Summer 2005

# Isoelectric point prediction from the amino acid sequence of a protein

Matthew Conte

THESIS

# ISOELECTRIC POINT PREDICTION FROM THE AMINO ACID SEQUENCE OF A PROTEIN

Submitted by
Matthew Conte
Department of Biological Sciences

In partial fulfillment of the requirements
For the Master of Science degree in
Bioinformatics at
Rochester Institute of Technology
Summer 2005

bioinformatics at RIT

**Rochester Institute of Technology**
**Department of Biological Sciences**
**Bioinformatics Program**

**To:** Head, Department of Biological Sciences

The undersigned state that Matthew A. Conte
(Student Name)

_____, a candidate for the Master of Science degree in
(Student Number)

Bioinformatics, has submitted his/her thesis and has satisfactorily defended it.

This completes the requirements for the Master of Science degree in Bioinformatics at Rochester Institute of Technology.

**Thesis committee members:**

| **Name** | **Date** |
|---|---|
| Gary R. Skuse (Committee Chair) | 31 Aug 2005 |
| Paul A. Craig (Thesis Advisor) | 8/31/05 |
| Name Illegible | Aug. 31, 2005 |
| Douglas P. Merrill | August 31, 2005 |
| | |

475-2532 (voice)
gary@bioinformatics.rit.edu

# Thesis/Dissertation Author Permission Statement

Title of thesis or dissertation: _____

_____

_____

_____

Name of author: Matthew Conte
Degree: Masters
Program: Bioinformatics
College: Science

I understand that I must submit a print copy of my thesis or dissertation to the RIT Archives, per current RIT guidelines for the completion of my degree. I hereby grant to the Rochester Institute of Technology and its agents the non-exclusive license to archive and make accessible my thesis or dissertation in whole or in part in all forms of media in perpetuity. I retain all other ownership rights to the copyright of the thesis or dissertation. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation.

***Print Reproduction Permission Granted:***

I, Matthew Conte, hereby **grant permission** to the Rochester Institute Technology to reproduce my print thesis or dissertation in whole or in part. Any reproduction will not be for commercial use or profit.

Signature of Author: Matthew Conte Date: 9-02-2005

***Print Reproduction Permission Denied:***

I, _____, hereby **deny permission** to the RIT Library of the Rochester Institute of Technology to reproduce my print thesis or dissertation in whole or in part.

Signature of Author: _____ Date: _____

***Inclusion in the RIT Digital Media Library Electronic Thesis & Dissertation (ETD) Archive***

I, _____, additionally grant to the Rochester Institute of Technology Digital Media Library (RIT DML) the non-exclusive license to archive and provide electronic access to my thesis or dissertation in whole or in part in all forms of media in perpetuity.

I understand that my work, in addition to its bibliographic record and abstract, will be available to the world-wide community of scholars and researchers through the RIT DML. I retain all other ownership rights to the copyright of the thesis or dissertation. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation. I am aware that the Rochester Institute of Technology does not require registration of copyright for ETDs.

I hereby certify that, if appropriate, I have obtained and attached written permission statements from the owners of each third party copyrighted matter to be included in my thesis or dissertation. I certify that the version I submitted is the same as that approved by my committee.

Signature of Author: _____ Date: _____

# Abstract

Proteins often do not migrate as expected in two dimensional electrophoresis based on their primary sequence. The predicted isoelectric point (pI) frequently does not coincide with experimental pI values obtained in the laboratory. The reasons for these differences led to this study. Initially, 2DE data from the *E. coli* proteome was collected and formatted. This dataset was split into three parts each consisting of different levels of pI discrepancy (ΔpI). The protein sequence data for each ΔpI subset was run through a pipeline. At each stage of the pipeline the data were analyzed by comparing each of the three ΔpI subsets to one another. The pipeline consisted of a naïve approach (considering individual amino acid frequencies), followed by the application four different alphabets to represent sequences in a simpler way by grouping similar amino acids based on their charge, functional, chemical, and hydrophobic properties . The final step in the pipeline involved investigating the dipeptides of all of these sequences using both the 20 amino acid alphabet and the simplified groupings. An evaluation of the alphabet dipeptide analysis demonstrated the existence of certain dipeptide sequences which correlate well with differences between predicted pI and experimental pI.

# Table of Contents

# Introduction

Two-dimensional gel electrophoresis (2DE) has been an important laboratory technique for the field of proteomics for over two decades. 2DE allows the researcher to separate and identify thousands of proteins from a cellular extract in a single experiment. 2DE is difficult and time consuming as it is necessary to determine ideal initial conditions, wait for results, and possibly change conditions after that ***(1)***. In addition, reproducibility of gels and comparison of 2DE results between separate groups has proved difficult ***(1)***. In 2DE, proteins are separated in the first dimension by their isoelectric points (the pH at which the net charge of the protein is zero) and in the second dimension by their molecular weights. The accurate prediction of protein isoelectric point (pI) and molecular weight (MW) using simply the amino acid sequence of the protein would be extremely valuable to researchers who use two-dimensional gel electrophoresis.

Computational procedures for calculating and predicting the pI from the amino acid composition of a protein based on the dissociation constants of the charged groups within the protein have been developed ***(2-8)***. The accuracy of these algorithms is limited by the certainty of the values for the dissociations constants and by microenvironmental effects such as charge-charge interactions and post-translational modifications.

To systematically explore the relationship between pI, molecular weight and protein sequence, a data set of proteins was collected and organized from a model organism. The *Escherichia coli* proteome was chosen since it contains few post-translational modifications such as methylation, acylation, gylcosylation, or

phosphorylation which can alter the pI/MW; the presence of these modifications makes pI/MW predictions much more difficult since the modifications in the proteins may cause them to migrate to a position on a 2-D gel that is quite different than what is predicted based solely on the amino acid sequence of the protein. *E. coli* is also one of the best characterized prokaryotes and much more data beyond simply the protein sequence for each protein is widely available for it.

At this point it is necessary to consider the basic structural features of proteins and the role of individual amino acids in the structure and function of proteins. Figure 1 below shows the structure of the 20 amino acids with side chain structures shown in red ***(10)***.

The charge on all proteins arises from some of the amino acid side chains, as well as the carboxy- and amino-termini, some prosthetic groups, and bound ions. Our pI prediction tool ***(11)*** is designed to calculate charge based on the side chains and carboxy- and amino-termini. The charge on amino acid side chains depends on the pH of the solution and the $pK_A$ of the side chains. It is also affected by the localized environment around a side chain. Our current calculation model uses the following $pK_A$ values for ionizable groups on the protein and does not make any adjustments to the $pK_A$ values of the side chains regardless of their environment within the protein (Table 1). We also assume that the separation is based on the total charge on the protein, not the mass-to-charge ratios.

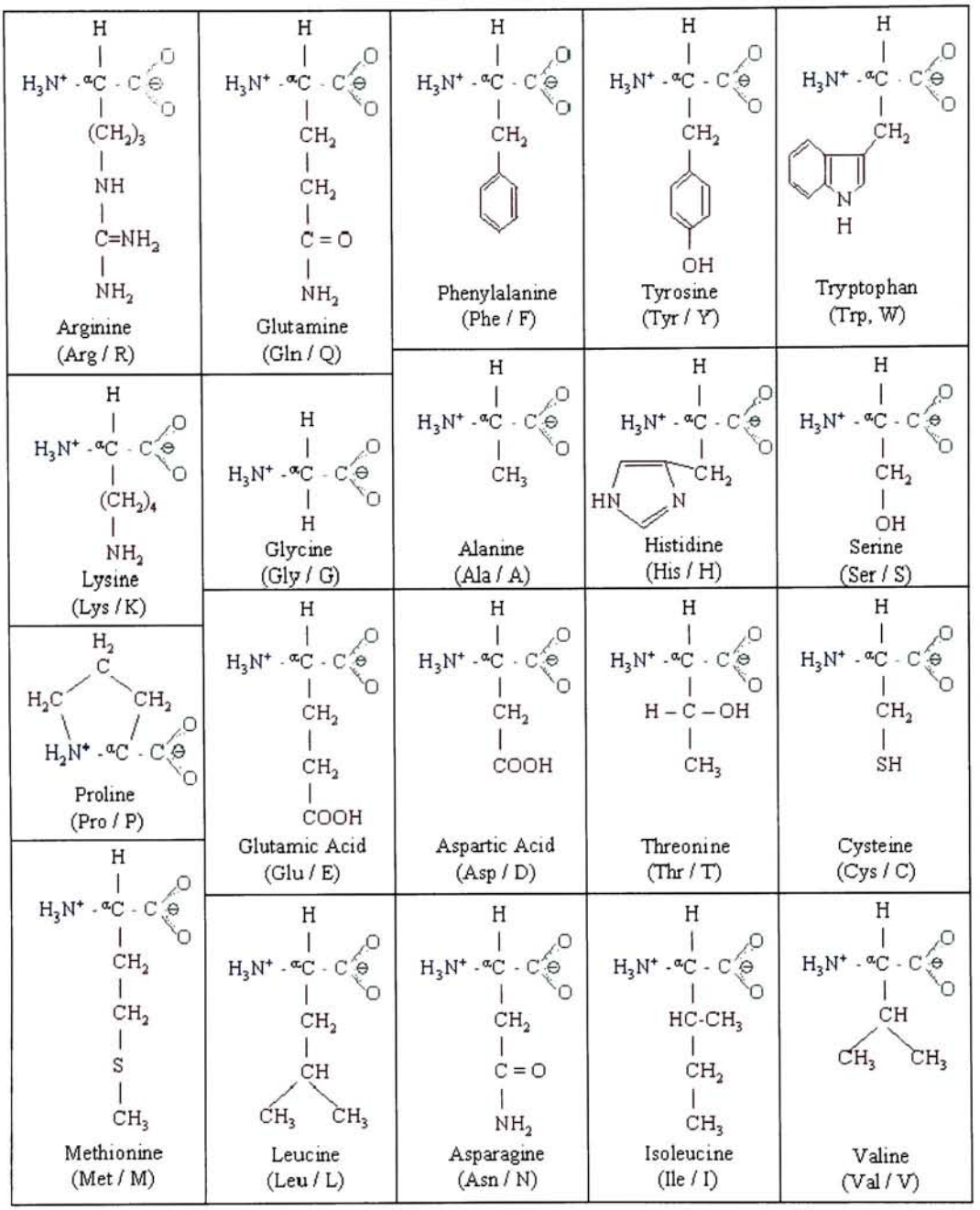


**Figure 1.** Structures of amino acids with side chains shown in red, carboxylate groups in green, and amino groups in blue ***(10)***.

The charge on the protein is the sum of the charges on the individual amino acid side chains. However, the charge on individual amino acid side chains can vary when they are near a group of non-polar or highly charged side chains. For example the normal

$pK_A$ for glutamic acid is about 4.1. In lysozyme, two glutamic acid residues are in the active site. One is in a polar environment and has a normal $pK_A$ value. The other glutamate side chain is in a hydrophobic environment, where a negative charge is energetically unfavorable. Therefore the $pK_A$ value for this glutamate side chain increases, which then decreases the extent of the deprotonation of that side chain. This is very important in the mechanism of lysozyme activity, which requires that one of the side chains be charged (deprotonated) and the other be uncharged (protonated) at the same time.

In a second example, the serine in the active sites of serine proteases has a much different acid-base behavior than other serines normally found in proteins ***(9)***. The normal $pK_A$ value for the hydroxyl group on the serine side chain is greater than 15, meaning that this group is not found in an ionized state in most proteins. In serine proteases, the interaction of the active site serine with nearby histidine and aspartate side chains (the so-called catalytic triad) leads to the ionization of the serine hydroxyl group. Meanwhile, the $pK_A$ value is reduced from about 15 to a value closer to 7 or 8. This example makes it clear that the microenvironment of an individual amino acid side chain can change it ionization behavior.

Other effects on the $pK_A$ of an amino acid side chain can be seen when certain amino acids are positioned next to each other. For example, a typical Arginine residue which is basic will have a $pK_A$ of about 12.5 (Table 1 below) and carry a full +1 charge in the physiological pH range. However, when two of these basic Arginine residues are adjacent in a protein sequence the $pK_A$ values will decrease, due to repulsion between the two positive charges. This reduction in $pK_A$ value, in turn, will cause one or both of the

arginine side chains to become less ionized and carry only a fractional positive charge. Table 1 below lists the typical $pK_A$ values for ionizable groups in proteins ***(9)***.

| Group | Typical $pK_a$ |
|---|---|
| Terminal α-carboxyl group | 3.1 |
| Aspartic acid, Glutamic acid | 4.1 |
| Histidine | 6.0 |
| Terminal α-amino group | 8.0 |
| Cysteine | 8.3 |
| Tyrosine | 10.9 |
| Lysine | 10.8 |
| Arginine | 12.5 |

**Table 1.** These are $pK_A$ values that are commonly found for these side chains when they are part of a protein. The $pK_A$ values for these side chains may be quite different for the free amino acid in solution. $pK_A$ values also depend on temperature, ionic strength, and the microenvironment of the ionizable group ***(9).***

As we began to consider the impact of amino acid sequence on ionization behavior of individual amino acid side chains, the need to create groups of amino acids based on their chemical and physical characteristics rather than concentrating on each individual amino acid became apparent. We elected to divide the amino acids into groups based on their chemical, functional, charge, and hydrophobic characteristics. Dividing sets of amino acids into these groups enables us to use smaller alphabets based on these characteristics as opposed to simply using the normal 20 letter amino acid alphabet in our calculations.

We used these property groups to rewrite a protein sequences into an alternative alphabet that is much smaller than the normal amino acid alphabet of 20 characters ***(12)***. Table 2 below describes how each different alphabet that was used is categorized based

on which amino acids fall under what particular types. The Methods section contains examples of protein sequences that have been translated into these different alphabets.

| Alphabet Type (size) | Code | Meaning | Amino Acids with that Code |
|---|---|---|---|
| Charge (3) | A | Negative | D, E |
| | C | Positive | H, K, R |
| | N | No charge | A,C,F,G,I,L,M, N,P,Q,S,T,V,W,Y |
| Chemical (8) | A | Acidic | D, E |
| | L | Aliphatic | A,G,I,L,V |
| | M | Amide | N,Q |
| | R | Aromatic | F,W,Y |
| | C | Basic | R,H,K |
| | H | Hydroxyl | S,T |
| | I | Imino | P |
| | S | Sulphur | C,M |
| Functional (4) | A | Acidic | D, E |
| | C | Basic | H, K, R |
| | H | Hydrophobic | A, F, I, L, M, P, V, W |
| | P | Polar | C, G, N, Q, S, T, Y |
| Hydrophobic (2) | I | Hydrophobic | A, F, I, L, M, P, V, W |
| | O | Hydrophilic | C, D, E, G, H, K, N, Q, R, S, T, Y |

**Table 2**. Description of four abbreviated amino acid sequence alphabets: Charge, Chemical, Functional, and Hydrophobic ***(12)***. Shown are the new alphabet codes used for each different alphabet, what each code represents in terms of properties of amino acids, and the specific amino acids that are included in each property.

Proteins that have a significant difference between their predicted pI/MW (obtained using similar algorithms as mentioned above) and their experimental pI/MW will be studied. As mentioned before, certain amino acids that occur in a particular succession need to be considered. These trends in the periodicity of certain amino acids

of certain proteins (those with large ΔpI values) that do not occur in the other proteins whose pI values were accurately predicted are important. They may lead to a more accurate prediction of the pI and MW all of proteins from their amino acid compositions.

## Methods

Forming the data set

The ExPASy Server's SWISS-2DPAGE database ***(13)*** provides extensive 2-D gel information for human, mouse, *Arabidopsis thaliana*, *Dictyostelium discoideum*, *E. coli*, *Saccharomyces cerevisiae*, and *Staphylococcus aureus (N315)* which are also cross-referenced in Swiss-Prot. Each protein in the database is collected and annotated from experimental 2-D gels read from reference maps. The database for this project contains 336 proteins of the *E. coli* proteome characterized by five different research groups ***(14-18)***. It was decided that the compilation of pI/MW sets for these proteins should be separated according to each research group since experimental conditions varied among them. The proteins contributed by the Phillips et al. ***(14)***, Pasquali et al. ***(15)***, and Vanbogelen et al. ***(16)*** groups were ignored because these proteins were also characterized by the Tonella et al. ***(17)***and Yan et al. ***(18)*** groups. Two sets were created; the first contains 228 of all the proteins denoted by Tonella et al. and 153 proteins of all the proteins denoted by Yan et al. The first set was also separated based on the pH range used for isoelectric focusing (pH 4-5, 4.5-5.5, 5-6, 5.5-6.7, 6-9, and 6-11). We concentrated on the Tonella et al. set because it covered more than 70% of the *E. coli* proteome and because all of the experiments were carried out under the same conditions.

We then matched the pI/MW data for each protein with its FASTA sequence. This allows us to compare experimental pI/MW values with predicted pI/MW values. ExPASy provides its own tool for predicting pI/MW which requires a list of Swiss-Prot protein IDs as its input of proteins ***(19)***. We have also developed our own tool that includes a pI/MW prediction which requires input of FASTA format sequences, Genbank format, or Protein Data Bank format ***(11)***. Both of these prediction tools are based (and especially pI for both tools) on a calculation using $pK_A$ values of amino acids as described earlier in the introduction and by Bjellqvist et al. ***(19)*** The first step was to retrieve the 2-D gel information for all of these proteins. ExPASy provides a way to get the data from each 2-D gel in a tab delimited format that includes each spot (one protein can have multiple spots on a gel). Having this data in a tab delimited format gave a far greater ease of use when later performing any type of analysis on the data (such as comparing experimental pI to predicted pI). The fields contained in these files included: gene name, protein description, SWISS-2DPAGE Serial Number, SWISS-2DPAGE Accession Number, identification method (gel matching, microsequencing, or peptide mass fingerprinting), experimental pI, experimental MW, and references.

A list of Swiss-Prot protein IDs (2DPAGE Accession Number – e.g. P00274) was then made for each of the gels. This list of proteins was then used to retrieve a FASTA file of the proteins from each gel (some proteins were repeated for multiple spots). The Swiss-Prot IDs were submitted to the NCBI tool for retrieving sequences at http://www.ncbi.nlm.nih.gov/entrez/batchentrez.cgi?db=Protein. The sequences were downloaded in FASTA format to be used in our prediction tool. Batch retrieval at NCBI was chosen over batch retrieval at ExPASy because the latter does not include, for

whatever reason, the initial methionine residue when retrieving in FASTA format. The FASTA file for the set of proteins from each gel was then fed into our tool where the output can be conveniently recorded to a Microsoft Excel file. However, problems occurred when using the FASTA file from NCBI in our tool since it would order the file based on Genbank accession number and not by Swiss-Prot ID which was needed to match the tab delimited file for each gel. This was solved by removing the Genbank accession number (leaving just the Swiss-Prot ID) from each protein entry in each respective FASTA file using a simple Perl script. This was facilitated by a few regular expressions most notably: ":%s/gi|\d*|sp|//" (quotations excluded). The pI/MW predict tool at ExPASy ***(19)*** was not quite as easy to use since it does not output into a format that can be imported into Excel readily. The output file was edited using the following regular expression: ":%s/\s\s*/\t/g" (quotations excluded) which transformed it into a tab delimited text file, allowing it to be easily manipulated in Excel. Nevertheless the "Compute pI/MW tool" at ExPASy ***(19)*** gave strikingly similar results to our tool.

Both experimental data sets derived from the Tonella data ***(17)*** and the Yan (DIGE) data ***(18)*** were compared with both pI/MW prediction tools and the results can be seen in the Excel files at http://www.rit.edu/~mac3948/E2D/Ecoli/.

Experimental and predicted pI values

Looking at the compiled data set it was noticeable that some predicted pI values were far different from experimental pI values. Some proteins differed in predicted pI versus experimental pI by as much as 1.86 pH units (e.g. P06128, Phosphate-binding periplasmic protein (PBP), see Appendix A). However, for other proteins the predicted

pI was exactly the same as the experimental pI (e.g. P06960, Ornithine carbamoyltransferase chain F (OTCase-2), see Appendix A).

To better characterize these discrepancies across all of the proteins a simple calculation was performed:

```
Experimental pI - predicted pI = Delta (Δ)pI          (Eq. 1)
```

The difference in experimental pI and predicted pI will be referred as ΔpI in this paper. The main focus of this project is to identify potential causes of varying ΔpI values.

The data set was then broken down into roughly thirds. The first subset of proteins consisted of 60 proteins where the ΔpI value was less than 0.1. Another subset held 58 proteins of ΔpI values greater than 0.3, but less than 0.7 ($0.3 < \Delta pI < 0.7$). The last third was put into a subset of 50 proteins where the ΔpI value was greater than 0.7. Refer to the tables in Appendix A for a list of the proteins in each ΔpI subset.

The following sections will provide the sequential steps that were performed on the analysis of these data subsets. It starts with a naïve approach to handling the data that deals with simply calculating raw frequencies of the 20 amino acids. The next section explains how we used the four different alphabets to analyze the data subsets, still focusing on individual amino acid frequencies. The dipeptide approaches are described next, followed by a final section that summarizes how the whole process flows together.

## Extracting useful information from collected subset sequences

### *Amino acid frequency analysis (the naïve approach)*

There is a naïve approach to finding a significant difference between each of the subsets of ΔpI ranges. This method involves determining the counts of each amino acid in each ΔpI subset and comparing the relative frequency of occurrence for each amino

acid between the ΔpI subsets. If a significant difference for any amino acid does exist between any of the ΔpI subsets, then this would be of great interest. It would then be possible to adjust a pI prediction algorithm based on individual amino acid frequency values and predict pI values that were closer to experimental values.

The first step in going about the naïve approach was to start from the list of proteins for each ΔpI subset. As previously described, the batch sequence retrieval at the NCBI was used to obtain a FASTA file that contained each sequence included in each ΔpI subset. A Perl program was then written to count the number of amino acids in each sequence from a FASTA file and calculate the frequency of each, outputting a tab delimited file displaying all of the frequencies for each sequence. The code of this program can be found in Appendix B – aacounts.pl.

Another Perl program was written which concatenates each separate sequence into one long sequence. This allows one to look at the amino acid frequencies encompassing each ΔpI subset as a whole instead of protein by protein. The program also makes sure that each protein sequence is kept separate and that the header line of each sequence is removed (see Appendix B – makeComposite.pl), which will be shown to be important shortly when looking at two amino acids that occur one right after the other (see *dipeptide approach*).

*Frequency of amino acids (alphabets approach)*

*Charge alphabet*

A more sophisticated analysis of amino acid frequency can be done if the amino acids are grouped according to the properties of their side chains. The structures of the

side chains of the amino acids can be used to assign them to four abbreviated amino acid alphabets (Charge, Chemical, Functional, and Hydrophobic). The Charge alphabet (see Table 2) is based on whether the side chain of an amino acid can have a positive or negative charge, or is simply uncharged (neutral). Glutamic Acid (Glu / E) and Aspartic Acid (Asp / D) are the only amino acids that contain the negatively charged carboxyl group ($COO^-$). Therefore, in the Charge alphabet they are grouped together and given the code **A**. Likewise, Lysine (Lys / K) and Arginine (Arg / R) are amino acids that contain the positively charged amino groups (the lysine side chain contains an ε-amino group and arginine has a guanidino group). In the Charge alphabet they are grouped together with the code **C**. Histidine (His / H) is also grouped into the positively charged amino acid group because protonation of the nitrogen on its side chain occurs easily. The remaining 15 amino acids have side chains which normally do not demonstrate charge behavior in proteins; they are grouped together and given the code **N**. An example of using the Charge alphabet can be seen below:

```
ACDEFGH    (original sequence)
   ↓
NNAANNC    (Charge alphabet sequence)
```

*Chemical alphabet*

The Chemical alphabet incorporates two groupings, acidic and basic with codes **A** and **C**, respectively. These groupings are analogous to the **A** and **C** groupings in the Charge alphabet for the same reasons. The Chemical alphabet characterizes the remaining 15 amino acids based on more than their lack of a charge. Asparagine (Asn / N) and Glutamine (Gln / Q) are amino acids that contain an amide ($CONH_2$) and are grouped together accordingly with the code **M**. Phenylalanine (Phe / F), Tryptophan (Trp

/ W), and Tyrosine (Tyr, Y) contain aromatic rings (code **R**). Serine (Ser / S) and Threonine (Thr / T) contain the hydroxyl group (OH) on their side chains (code **H**). Proline (Pro / P) contains an imino group (>C=NH) on its side chain (code **I**). Finally, the sulfur containing amino acids are Cysteine (Cys / C) and Methionine (Met / M) are grouped together with code **S**. An example of using the Chemical alphabet can be seen below:

```
ACDEFGHNPS      (original sequence)
    ↓
LSAARACMIH      (Chemical alphabet sequence)
```

*Functional alphabet*

The Functional alphabet again incorporates the **A** (acidic) and **C** (basic) groups as did the Charge and Chemical alphabets. The Functional alphabet characterizes the remaining amino acids into 2 groups: **H** (hydrophobic) and **P** (polar) based on whether the amino acid is hydrophobic (such as Alanine) or polar (such as Cysteine). An example of using the Functional alphabet can be seen below:

```
ACDEFGH    (original sequence)
   ↓
HPAAHPC    (Functional alphabet sequence)
```

*Hydrophobic alphabet*

The Hydrophobic alphabet is similar to the latter half of the Functional alphabet. It groups amino acids based only on hydrophobicity. Amino acids that are hydrophilic (such as Cysteine) are given the code **I**. Amino acids that are hydrophobic (such as Alanine) are given the code **O**. An example of using the Hydrophobic alphabet can be seen below:

```
ACDEFGH    (original sequence)
   ↓
OIIIOII    (Hydrophobic alphabet sequence)
```

Perl programs were written that convert normal sequences into each of the four alphabets just described (see charge.pl, chemical.pl, functional.pl, and hydro.pl in Appendix B). The programs also calculate and display the frequency of each alphabetic code that is chosen.

*Frequency of amino acids (dipeptide approach)*

The problem that certain abnormal $pK_A$ side chains values of amino acids affecting the overall charge of a protein still had not been dealt with up until this point. All that had been considered was the sum of a set of strict $pK_A$ values for each amino acid without taking into account any changes that might occur due to certain amino acids being next to other amino acids in sequence. The approach to solving this problem was to examine every "dipeptide" in the three ΔpI subsets. A sequence of length 7 has 6 dipeptides. For example,

| Sequence: | Dipeptides: | Dipeptide counts: | Frequency: |
|---|---|---|---|
| ABCABBC | AB | AB = 2 | 0.333 |
| | BC | BC = 2 | 0.333 |
| | CA | CA = 1 | 0.167 |
| | AB | BB = 1 | 0.167 |
| | BB | | |
| | BC | | |

The frequency at which each dipeptide occurs in a particular sequence is of interest, particularly, when they are considered in each ΔpI subset. A Perl program was written that counts each dipeptide in a sequence and displays the frequency of each

dipeptide in the sequences of the FASTA a file that is input (see Appendix B - dipeps.pl for dipeptides output in increasing order or dipepsA.pl for dipeptides output alphabetically from AA ... VV). As was the case earlier with the normal amino acid alphabet, the number of different dipeptides (20 x 20 = 400 for the normal alphabet) became problematic. The same dipeptide technique was applied to sequences after converting them into the Charge, Chemical, Functional, and Hydrophobic alphabets to alleviate this problem.

Combining an entire ΔpI subset of FASTA sequences into one long sequence (using makeComposite.pl – see Appendix B) also became problematic. To count the number of dipeptides in a *set* of sequences that has been combined into one long sequence, special attention needs to be paid so that the last amino acid in one sequence and the first amino acid in the next sequence are not counted as a dipeptide. The format of the output file from makeComposite.pl handles this problem by replacing each accession line with a blank new line. The other programs can now use this formatted FASTA file so that the dipeptide counts are just as accurate as naïve and alphabet counts.

*Pipeline Workflow*

So far there have been stages at which the frequency of an amino acid, group of amino acids (coded according to the four alphabets), dipeptide, or grouped dipeptide (coded according to the four alphabets) has been examined. The process of transforming the data to reach each of these stages may appear somewhat confusing. Figure 2 below diagrams how to go from an initial set of FASTA sequences (for each ΔpI subset) to each stage of analysis. The flow in taking the naïve approach would go from FASTA

sequence to makeComposite.pl to aacounts.pl and then analysis. However, the flow for examining dipeptides with a functional alphabet is more complex. It begins by transferring the FASTA sequence to makeComposite.pl to functional.pl to dipeps.pl (or dipepsA.pl) followed by analysis. Table 3 below gives a brief description of each program used in this pipeline workflow (for a more detailed description and code of each program see Appendix B).

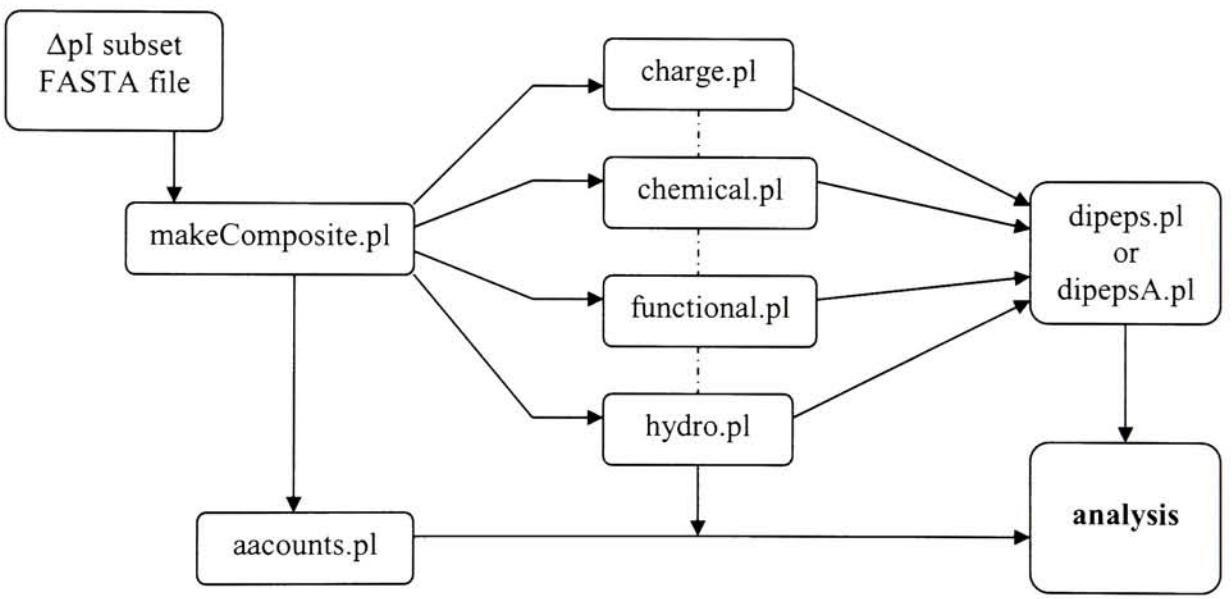


**Figure 2.** Workflow diagram that shows how to get to each stage of analysis (naïve, alphabets, dipeptides).

| Program | Description |
|---|---|
| aacounts.pl | Counts the number of each amino acid (normal alphabet) in a sequence from a FASTA file and determines the frequency of each. Output is to FASTAfilename.aacounts |
| charge.pl | Converts the amino acids from the sequences in a FASTA file into a 3-letter alphabet using the charge() method in Bio::Tools::OddCodes ***(12)***. It then counts the number of each code for each sequence as well as each frequency. |
| chemical.pl | Converts the amino acids from the sequences in a FASTA file into an 8-letter alphabet using the chemical() method in Bio::Tools::OddCodes ***(12)***. It then counts the number of each code for each sequence as well as each frequency. |
| dipeps.pl | Counts the number of each different amino acid pair for each sequence in the given FASTA files. It displays each pair in order from highest frequency to lowest. |
| dipepsA.pl | Counts the number of each different amino acid pair for each sequence in the given FASTA files. It displays each pair in alphabetical order (AA ... VV). |
| functional.pl | Converts the amino acids from the sequences in a FASTA file into a 4-letter alphabet using the functional() method in Bio::Tools::OddCodes ***(12)***. It then counts the number of each code for each sequence as well as each frequency. |
| hydro.pl | Converts the amino acids from the sequences in a FASTA file into a 2-letter alphabet using the hydrophobic() method in Bio::Tools::OddCodes ***(12)***. It then counts the number of each code for each sequence as well as each frequency. |
| makeComposite.pl | Converts FASTA files of multiple sequences into a single (composite) sequence. This composite sequence is then able to be used with other programs listed here. |

**Table 3.** Description of the programs used in this pipeline workflow. Appendix B provides a longer description and the source code for each program.

# Results

*Naïve approach*

The intitial naïve approach to analyzing the data set was done to determine the counts of each amino acid (using the normal alphabet) in each ΔpI subset (ΔpI < 0.1; 0.3 < ΔpI < 0.7; ΔpI > 0.7) and compare the relative frequency of occurrence for each amino acid between the ΔpI subsets. A comparison of the frequencies between the ΔpI < 0.1 subset and the 0.3 < ΔpI < 0.7 subset is shown in Figure 3. A similar comparison between the ΔpI < 0.1 subset and the ΔpI > 0.7 subset is displayed in Figure 4.

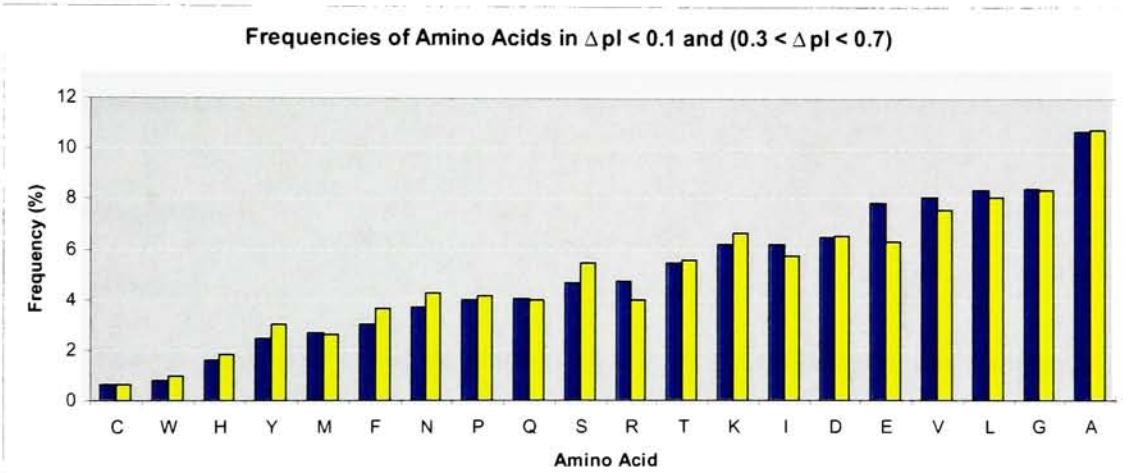


**Figure 3.** Frequency of Individual Amino Acids in Two ΔpI Subsets. The X axis labels represent the one letter abbreviations of the amino acids. Shown in blue are is the ΔpI < 0.1 subset and shown in yellow is the 0.3 < ΔpI < 0.7 subset. The ΔpI < 0.1 subset consists of 60 proteins which comprise 22472 total amino acids. The 0.3 < ΔpI < 0.7 subset consists of 58 proteins which comprise 17906 total amino acids. More information about each individual protein in these ΔpI subsets can be seen in Appendix A.

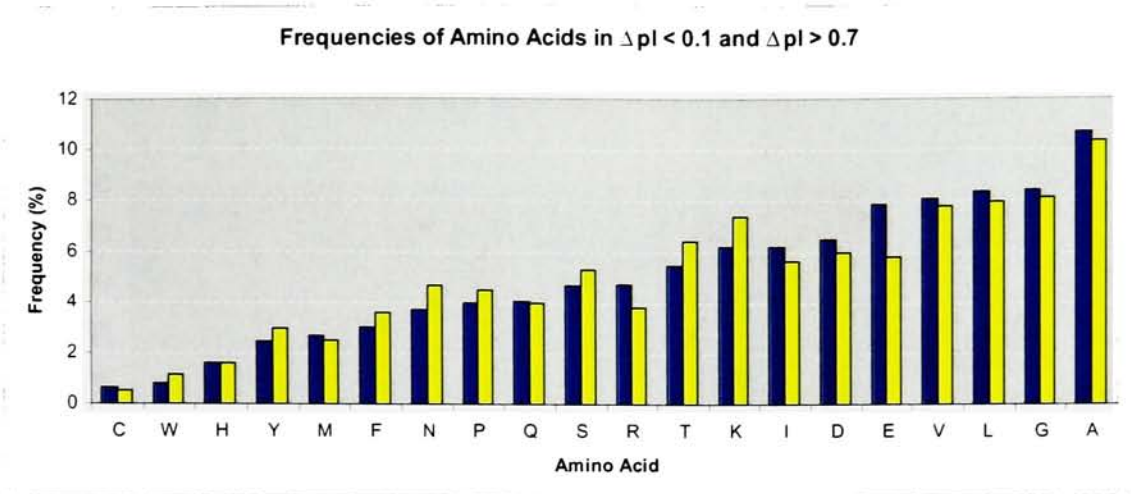


**Figure 4.** Frequency of Individual Amino Acids in Two ΔpI Subsets. The X axis labels represent the one letter abbreviations of the amino acids. Shown in blue are is the ΔpI < 0.1 subset and shown in yellow is the ΔpI > 0.7 subset. The ΔpI < 0.1 subset consists of 60 proteins which comprise 22472 total amino acids. The ΔpI > 0.7 subset consists of 50 proteins which comprise 15581 total amino acids. More information about each individual protein in these ΔpI subsets can be seen in Appendix A.

*Alphabets approach*

*-Charge*

The next step in analysis was to convert each of the ΔpI subsets into a sequence that utilizes the four alphabets. This decreases the size of the amino acid alphabet and reduces the number of variables being examined. The different alphabets are summarized in Table 2. Using the Charge alphabet, a comparison of the frequencies between the ΔpI < 0.1 subset and the 0.3 < ΔpI < 0.7 subset is shown in Figure 5. Again using the Charge alphabet a similar comparison between the ΔpI < 0.1 subset and the ΔpI > 0.7 subset is displayed in Figure 6.

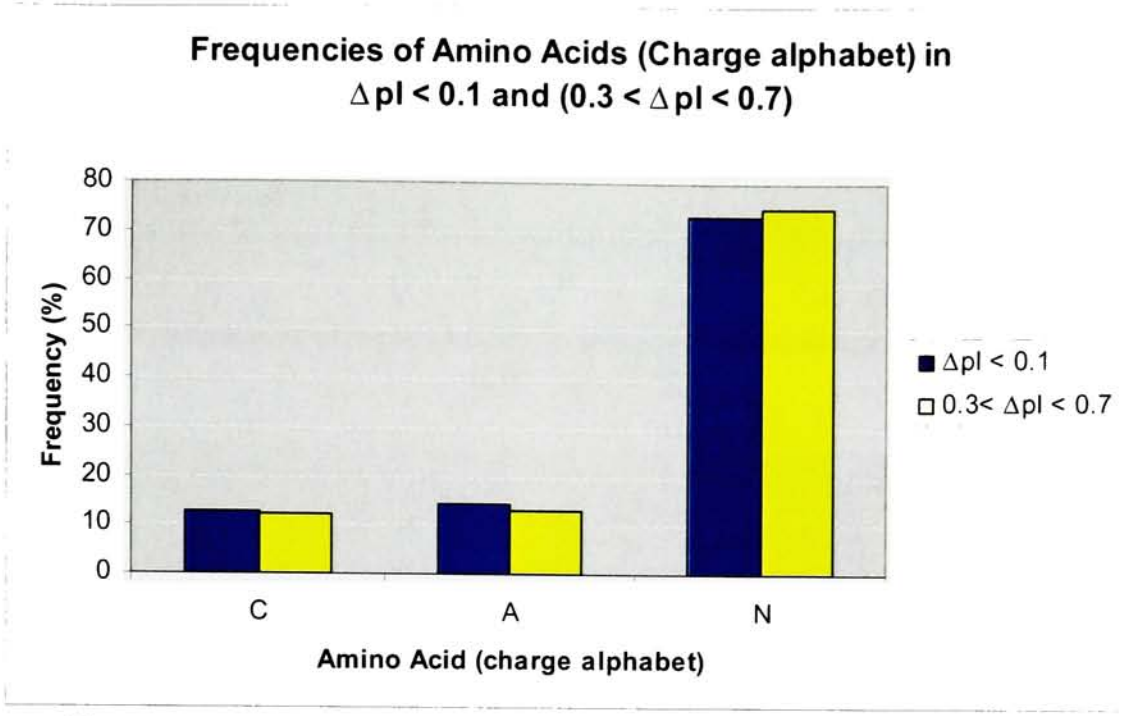


**Figure 5.** Frequency of Amino Acids Using the Charge Alphabet in Two ΔpI Subsets.

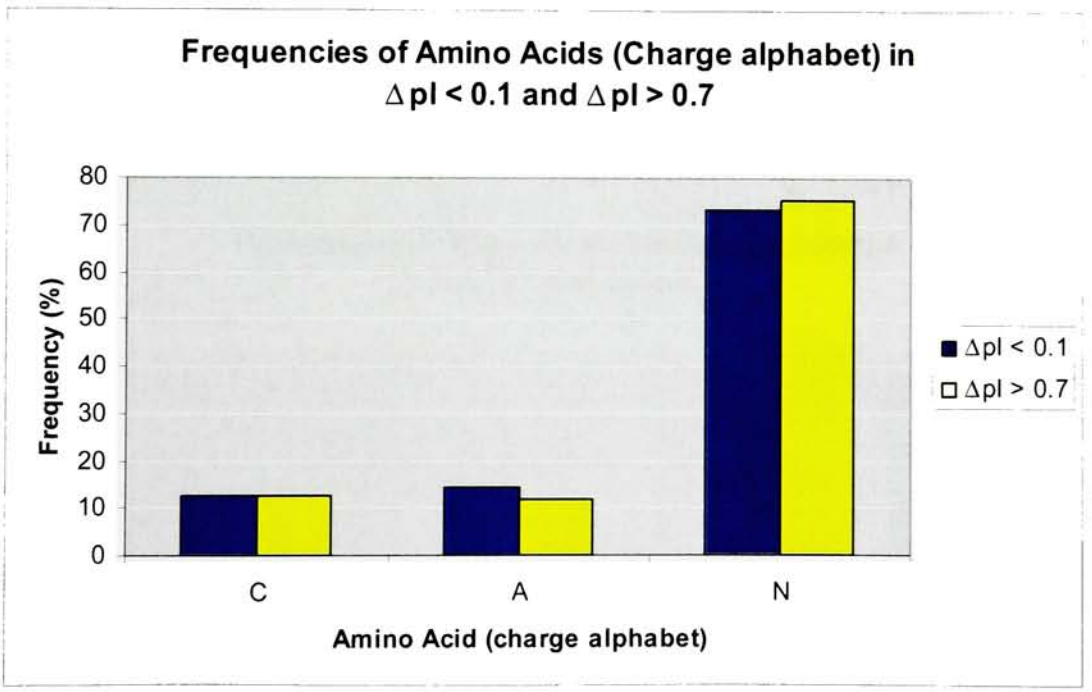


**Figure 6.** Frequency of Amino Acids Using the Charge Alphabet in Two ΔpI Subsets.

*-Chemical*

Using the Chemical alphabet, a comparison of the frequencies between the $\Delta pI < 0.1$ subset and the $0.3 < \Delta pI < 0.7$ subset is shown in Figure 7. Figure 8 displays the same comparison between the $\Delta pI < 0.1$ subset and the $\Delta pI > 0.7$ subset.

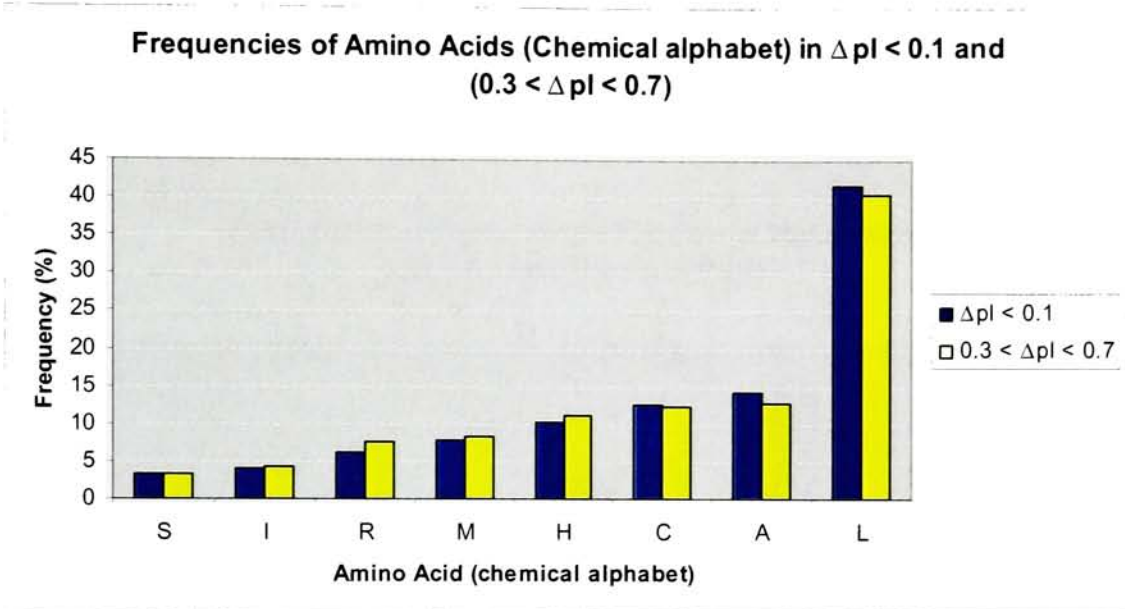


**Figure 7.** Frequency of Amino Acids Using the Chemical Alphabet in Two ΔpI Subsets.

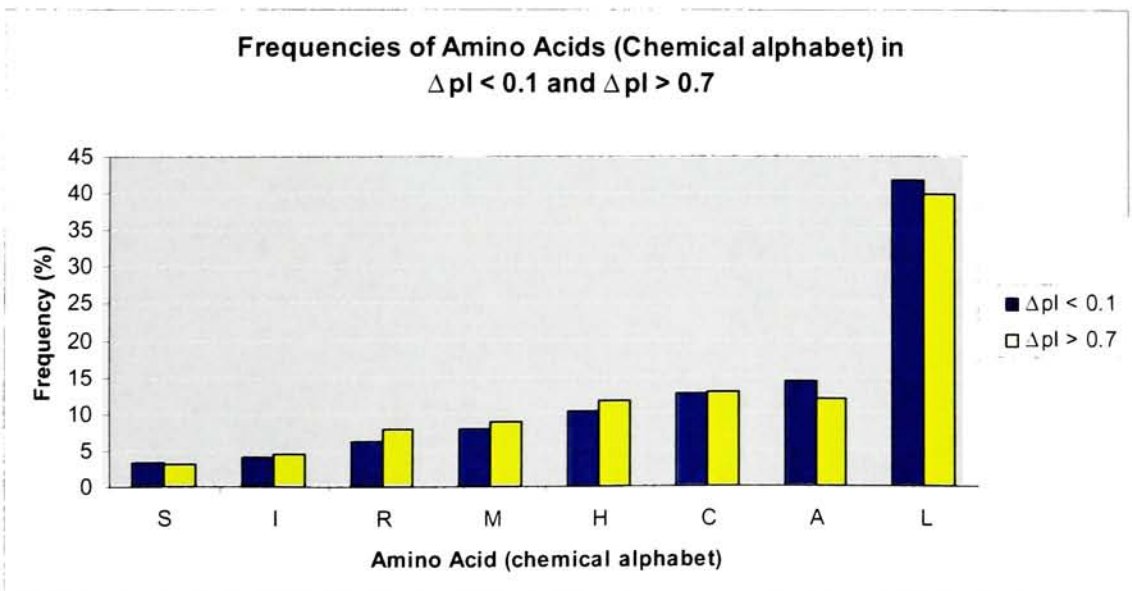


**Figure 8.** Frequency of Amino Acids Using the Chemical Alphabet in Two ΔpI Subsets.

*-Functional*

Using the Functional alphabet, a comparison of the frequencies between the $\Delta pI < 0.1$ subset and the $0.3 < \Delta pI < 0.7$ subset is shown in Figure 9. Again using the Functional alphabet a similar comparison between the $\Delta pI < 0.1$ subset and the $\Delta pI > 0.7$ subset is displayed in Figure 10.

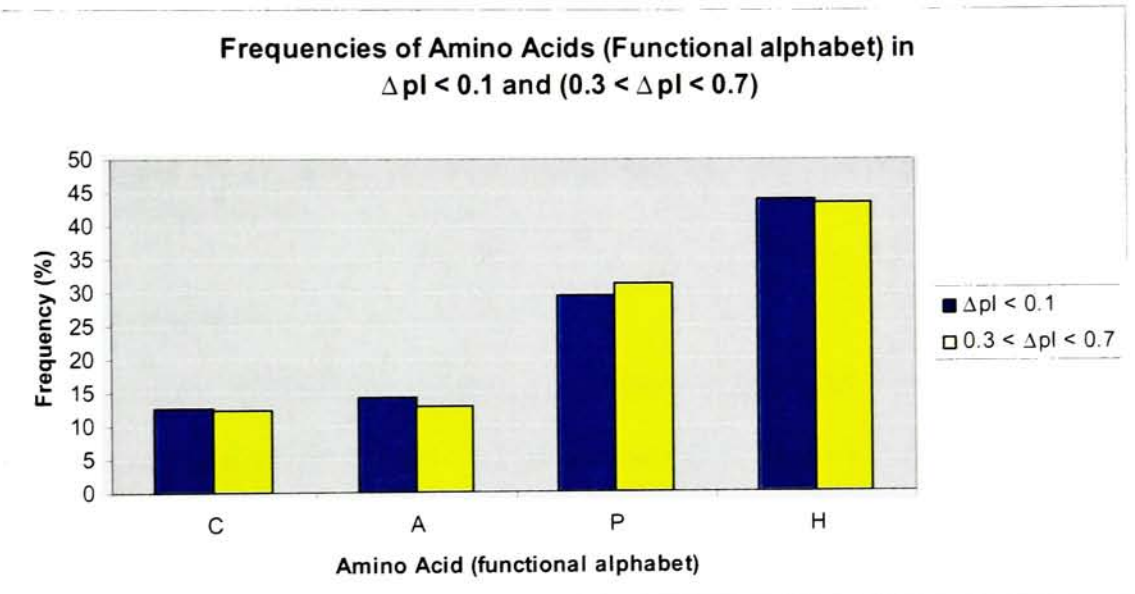


**Figure 9.** Frequency of Amino Acids Using the Functional Alphabet in Two ΔpI Subsets.

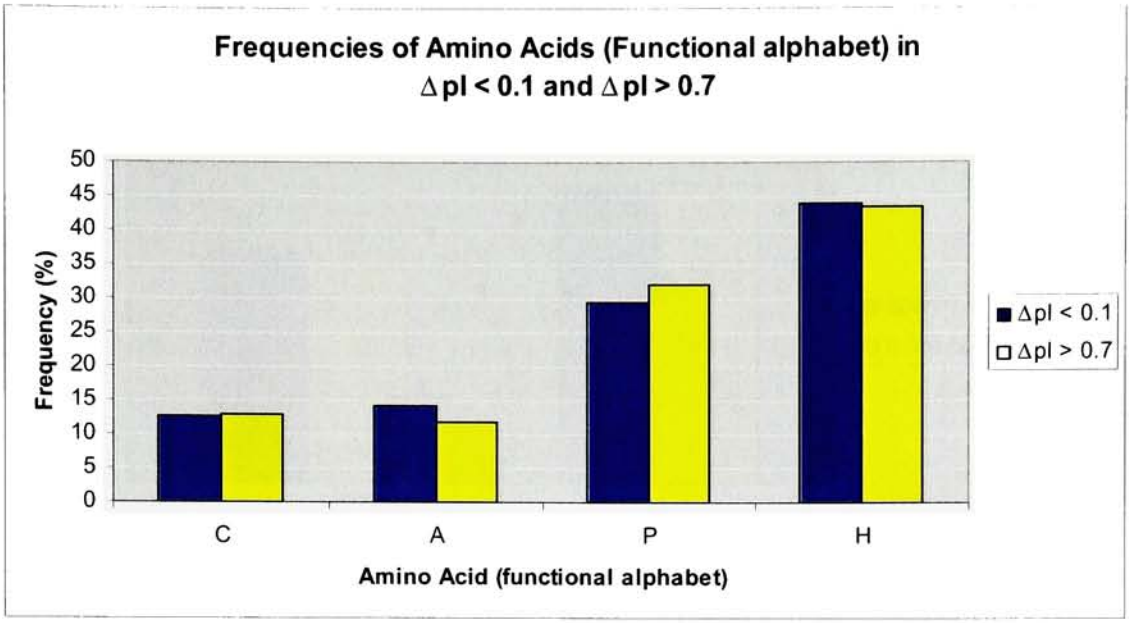


**Figure 10.** Frequency of Amino Acids Using the Functional Alphabet in Two ΔpI Subsets.

*-Hydrophobic*

Using the Hydrophobic alphabet, a comparison of the frequencies between the $\Delta pI < 0.1$ subset and the $0.3 < \Delta pI < 0.7$ subset is shown in Figure 11. Again using the Hydrophobic alphabet a similar comparison between the $\Delta pI < 0.1$ subset and the $\Delta pI > 0.7$ subset is displayed in Figure 12.

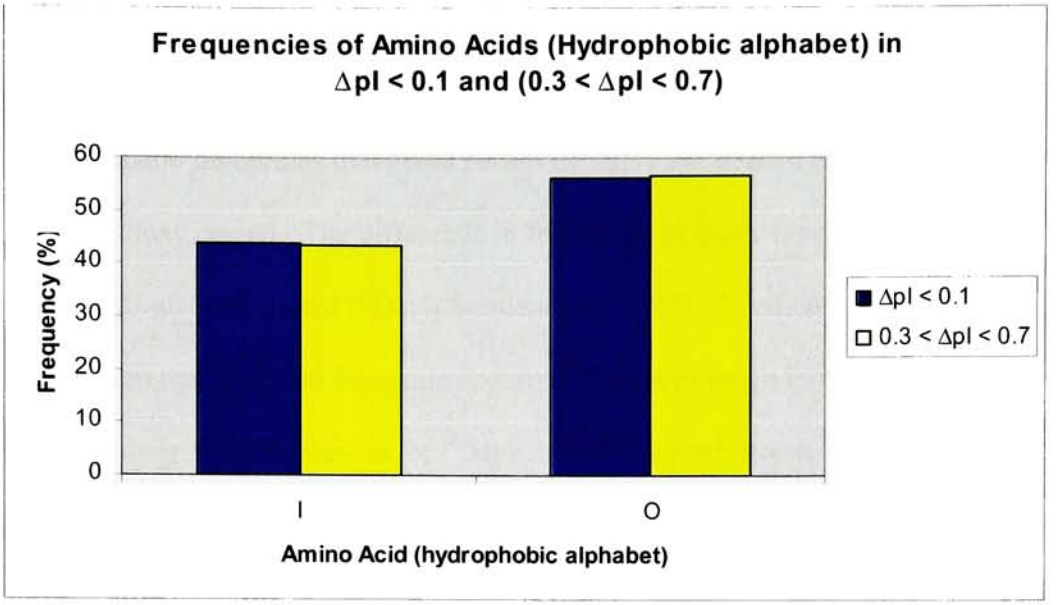


**Figure 11.** Frequency of Amino Acids Using the Hydrophobic Alphabet in Two ΔpI Subsets.

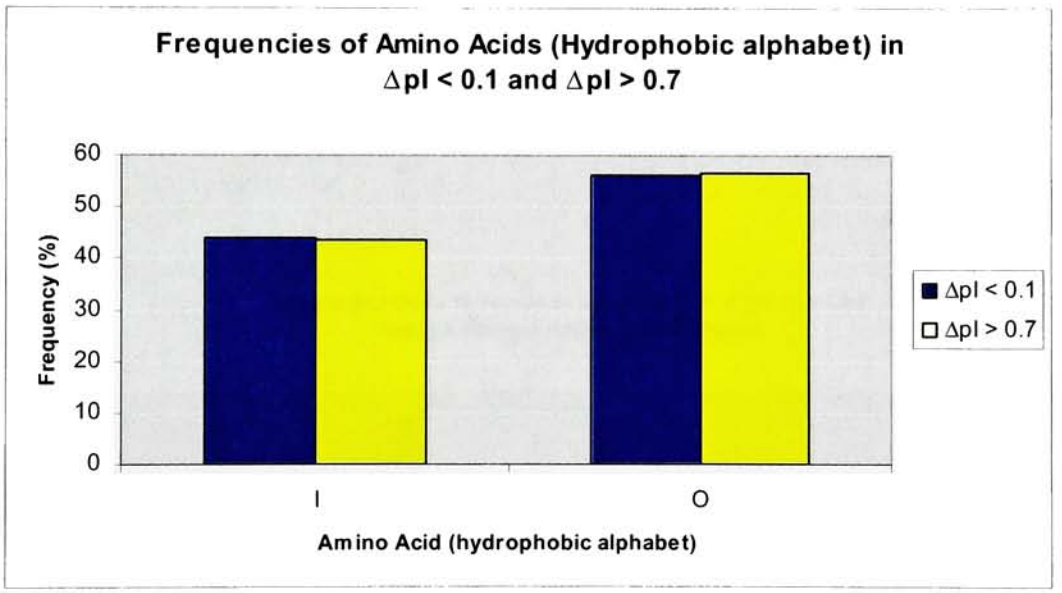


**Figure 12.** Frequency of Amino Acids Using the Hydrophobic Alphabet in Two ΔpI Subsets.

*Dipeptide approach*

Using a more sophisticated method that looks at dipeptides of a sequence gave an entirely new set of results. The first way of looking at dipeptides of the three ΔpI subsets

is similar to the naïve approach in that it just examines dipeptides using the normal amino acid alphabet. This results in upwards of 400 different dipeptides (there may be slightly fewer than 400 dipeptides in a given subset owing to the chance that not all possible dipeptides may occur). The difference in frequency of every dipeptide between ΔpI subsets was also calculated ("Delta frequency" or "%"). In other words, a Delta % of 100 would mean that a certain dipeptide occurred 2 times as much in one subset compared to another subset. The differences, or "Delta %" values can be seen in Figure 13 when comparing the $\Delta pI < 0.1$ subset and the $0.3 < \Delta pI < 0.7$ subset. Figure 14 shows the similar Delta % values when comparing the $\Delta pI < 0.1$ subset and the $\Delta pI > 0.7$ subset. To better explain Figures 13-16, consider the bar indicated by the arrow in Figure 13. This bar represents the 11 times that there was a Δ% value between 100% and 150% when comparing dipeptide frequencies in the two different ΔpI sets.

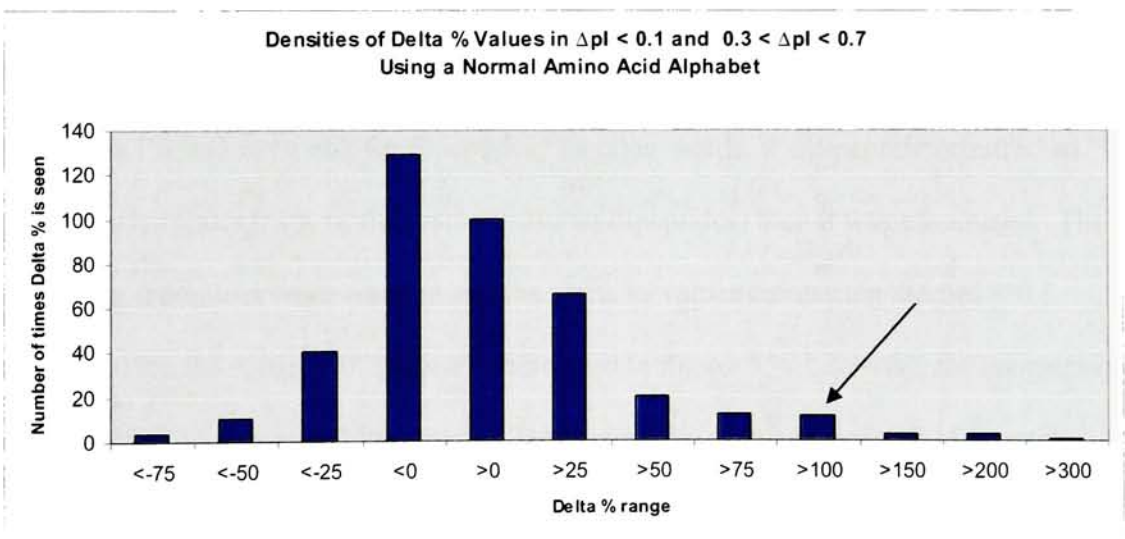


**Figure 13.** Density of Delta % Values of Dipeptides in Two ΔpI Subsets. The $\Delta pI < 0.1$ subset consists of 60 proteins which comprise 22412 total dipeptides. The $0.3 < \Delta pI < 0.7$ subset consists of 58 proteins which comprise 17848 total dipeptides. More information about each individual protein in these ΔpI subsets can be seen in Appendix A.

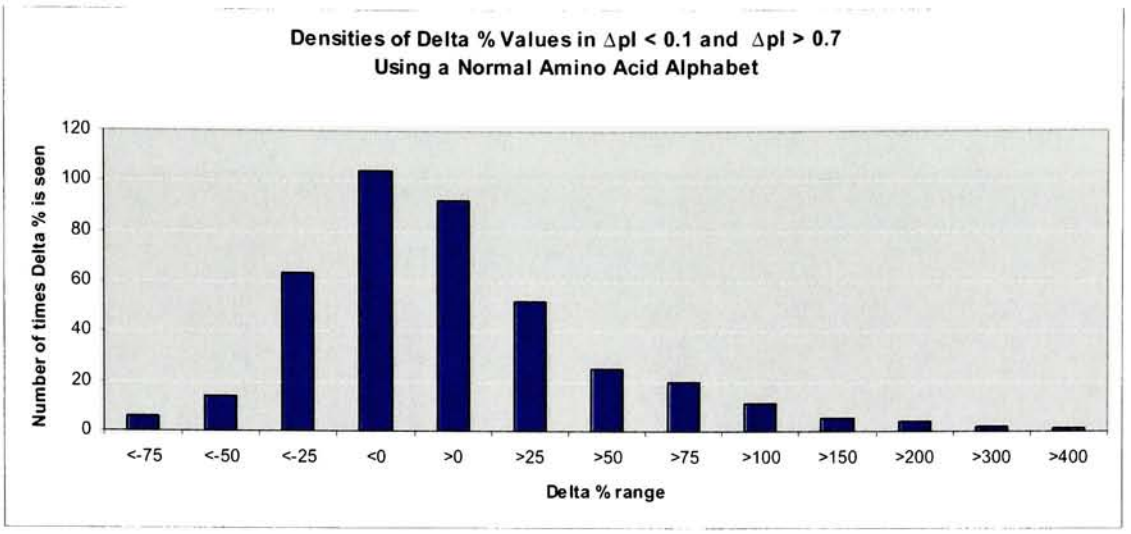


**Figure 14.** Density of Delta % Values of Dipeptides in Two ΔpI Subsets. The ΔpI < 0.1 subset consists of 60 proteins which comprise 22412 total dipeptides. The ΔpI > 0.7 subset consists of 50 proteins which comprise 15531 total dipeptides. More information about each individual protein in these ΔpI subsets can be seen in Appendix A.

*Dipeptide Threshold*

A similar analysis was performed on the same ΔpI subsets where dipeptides that had a very low frequency (which may change its Delta % value too rapidly, see Discussion for an elaboration) were monitored. A frequency of occurrence threshold value of 0.1% had to be met for dipeptides. In other words, if a dipeptide occurred so infrequently (under 0.1% of the total number of dipeptides) then it was eliminated. The remaining dipeptides were counted and the Delta % values comparing the ΔpI < 0.1 subset and the 0.3 < ΔpI < 0.7 subset can be seen in Figure 15. Likewise, the comparison for the ΔpI < 0.1 subset and the ΔpI > 0.7 subset can be seen in Figure 16. Dipeptides that were found in the extreme positive or negative ranges of these figures are indicated by the one letter amino acid codes. For instance, the dipeptide RR (arginine-arginine) in Figure 15 was found much less frequently in the ΔpI < 0.1 dataset than in the 0.3 < ΔpI < 0.7 dataset.

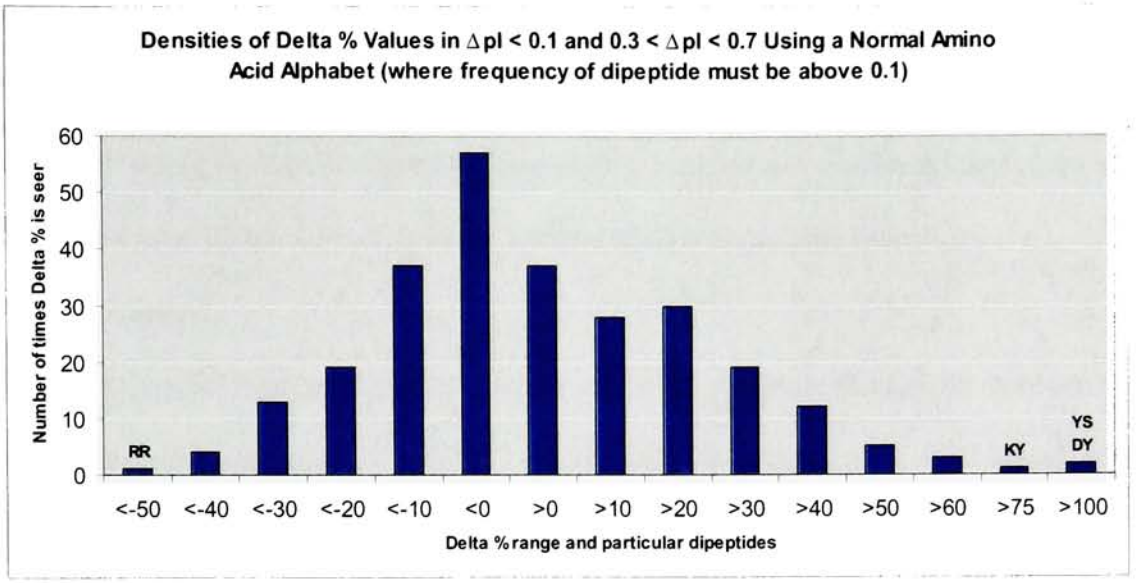


**Figure 15.** Density of Delta % Values of Dipeptides in Two ΔpI Subsets with a Threshold of 0.1%.

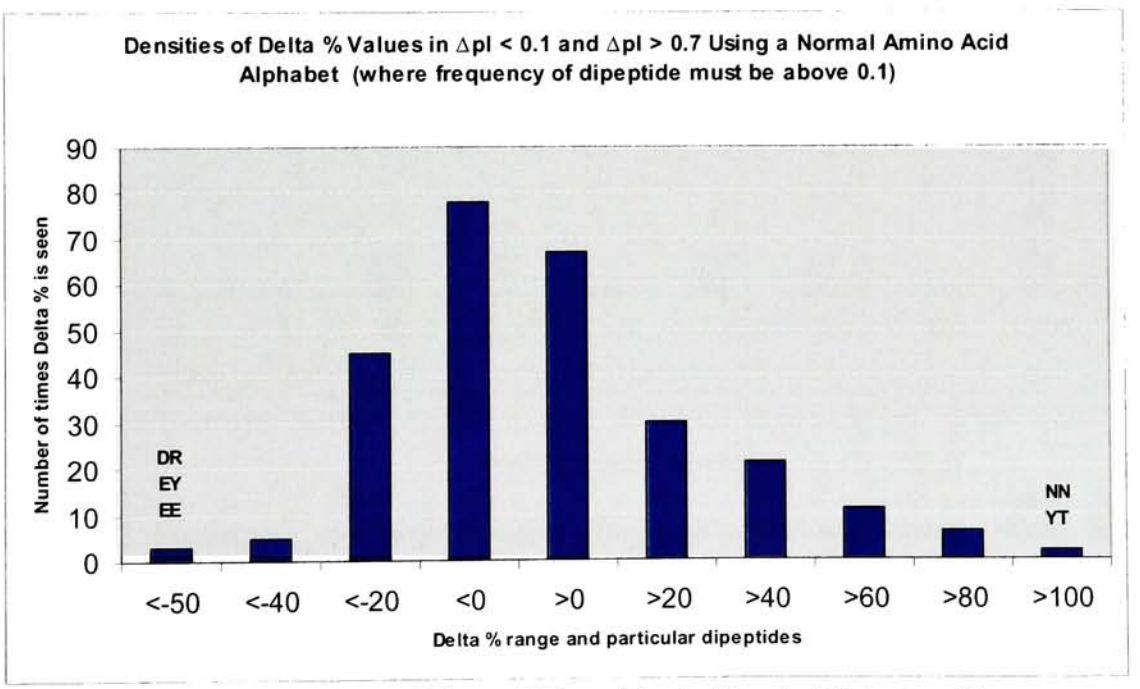


**Figure 16.** Density of Delta % Values of Dipeptides in Two ΔpI Subsets with a Threshold of 0.1%.

*Dipeptide using Alphabets*

The final step in analysis was to combine the alphabet and dipeptide approaches together. Using the smaller alphabets dramatically reduced and condensed the results as compared to using the normal alphabet which creates 400 possible dipeptides.

*-Charge*

Using the Charge alphabet, a comparison of the dipeptide frequencies between the $\Delta pI < 0.1$ subset and the $0.3 < \Delta pI < 0.7$ subset is shown in Figure 17 as well as the Delta % values for each dipeptide. The same comparison is shown between the $\Delta pI < 0.1$ subset and the $\Delta pI > 0.7$ subset in Figure 18.

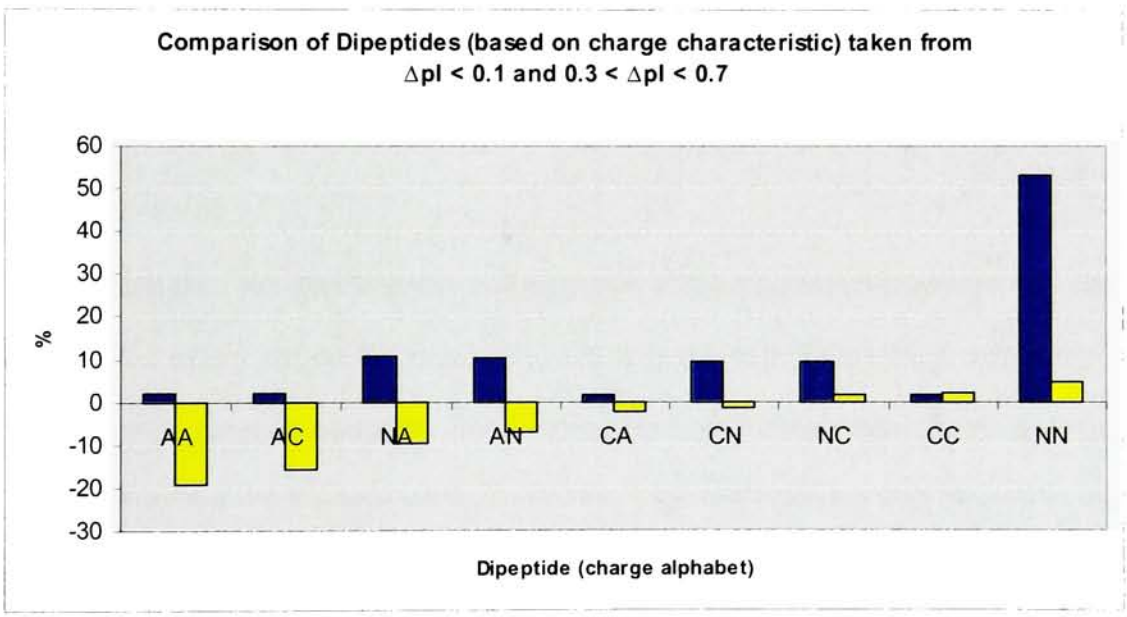


**Figure 17.** Frequencies of Charge Alphabet Dipeptides in Two ΔpI Subsets. Shown in blue are the frequencies of each dipeptide in the $\Delta pI < 0.1$ subset and shown in yellow is difference in frequency for each didpeptide between the $\Delta pI < 0.1$ subset and the $0.3 < \Delta pI < 0.7$ subset.

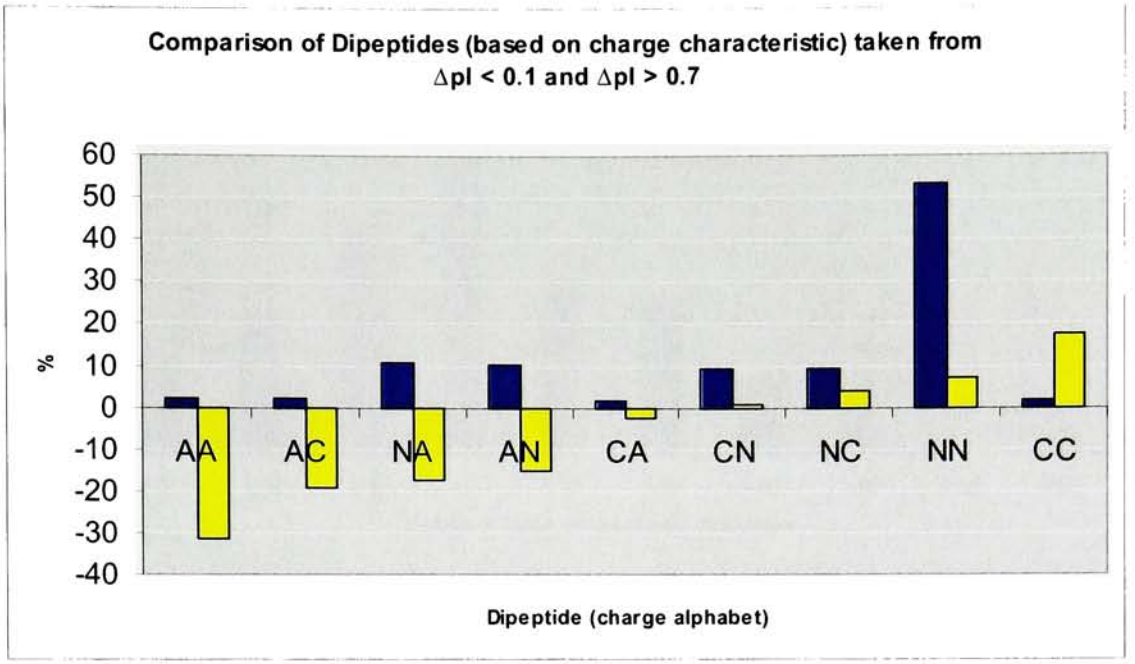


**Figure 18.** Frequencies of Charge Alphabet Dipeptides in Two ΔpI Subsets. Shown in blue are the frequencies of each dipeptide in the ΔpI < 0.1 subset and shown in yellow is difference in frequency for each didpeptide between the ΔpI < 0.1 subset and the ΔpI > 0.7 subset.

*-Chemical*

Using the Chemical alphabet, a comparison of the dipeptide frequencies between the ΔpI < 0.1 subset and the 0.3 < ΔpI < 0.7 subset is shown in Figure 19 as well as the Delta % values for each dipeptide. The same comparison is shown between the ΔpI < 0.1 subset and the ΔpI > 0.7 subset in Figure 20. The Chemical alphabet with dipeptides was sufficiently large that it was not possible to display all the possible dipeptide combinations in Figures 19 and 20. Instead only the density values were chosen to display.

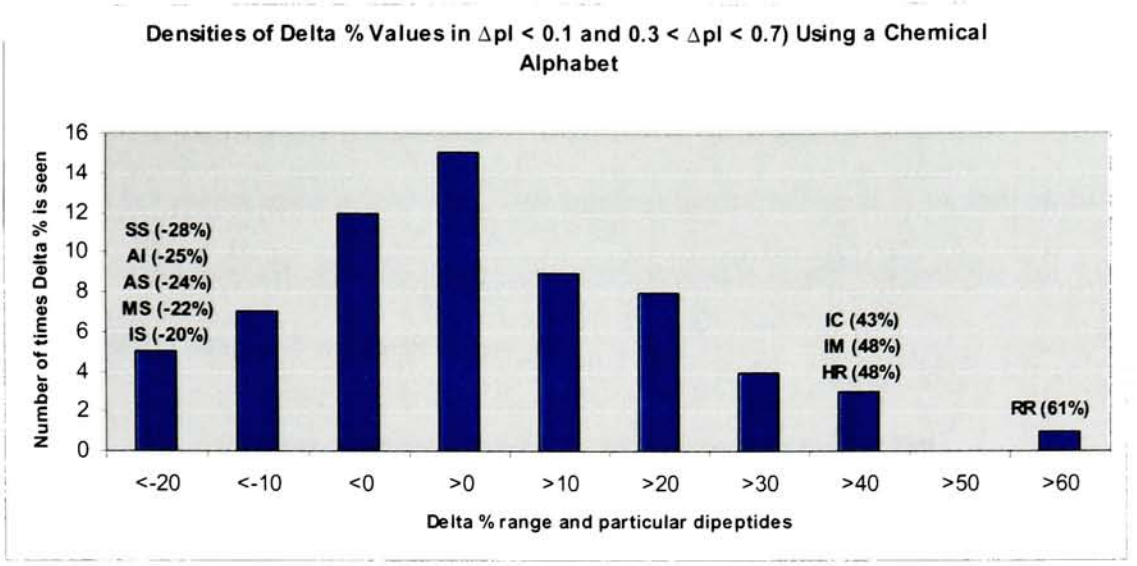


**Figure 19.** Density of Delta % Values of Chemical Alphabet Dipeptides in Two ΔpI Subsets. The ΔpI < 0.1 subset consists of 60 proteins which comprise 22412 total dipeptides. The 0.3 < ΔpI < 0.7 subset consists of 58 proteins which comprise 17848 total dipeptides. More information about each individual protein in these ΔpI subsets can be seen in Appendix A.

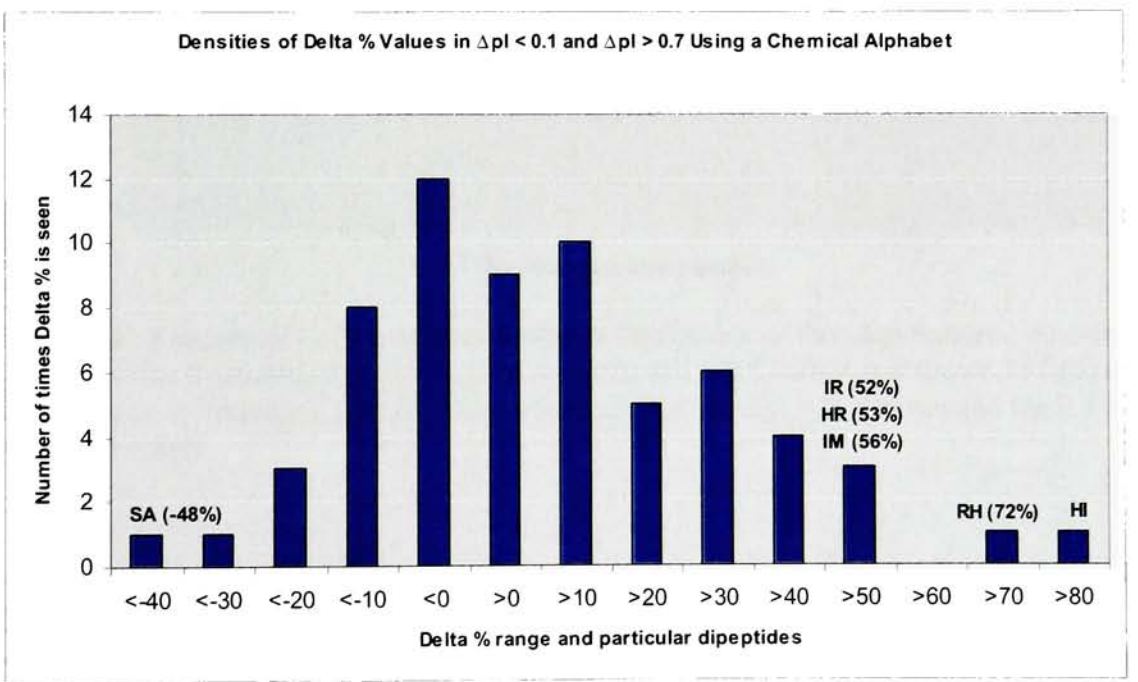


**Figure 20.** Density of Delta % Values of Chemical Alphabet Dipeptides in Two ΔpI Subsets. The ΔpI < 0.1 subset consists of 60 proteins which comprise 22412 total dipeptides. The ΔpI > 0.7 subset consists of 50 proteins which comprise 15531 total dipeptides. More information about each individual protein in these ΔpI subsets can be seen in Appendix A.

*-Functional*

Using the Functional alphabet, a comparison of the dipeptide frequencies between the $\Delta pI < 0.1$ subset and the $0.3 < \Delta pI < 0.7$ subset is shown in Figure 21 as well as the Delta % values for each dipeptide. The same comparison is shown between the $\Delta pI < 0.1$ subset and the $\Delta pI > 0.7$ subset in Figure 22.

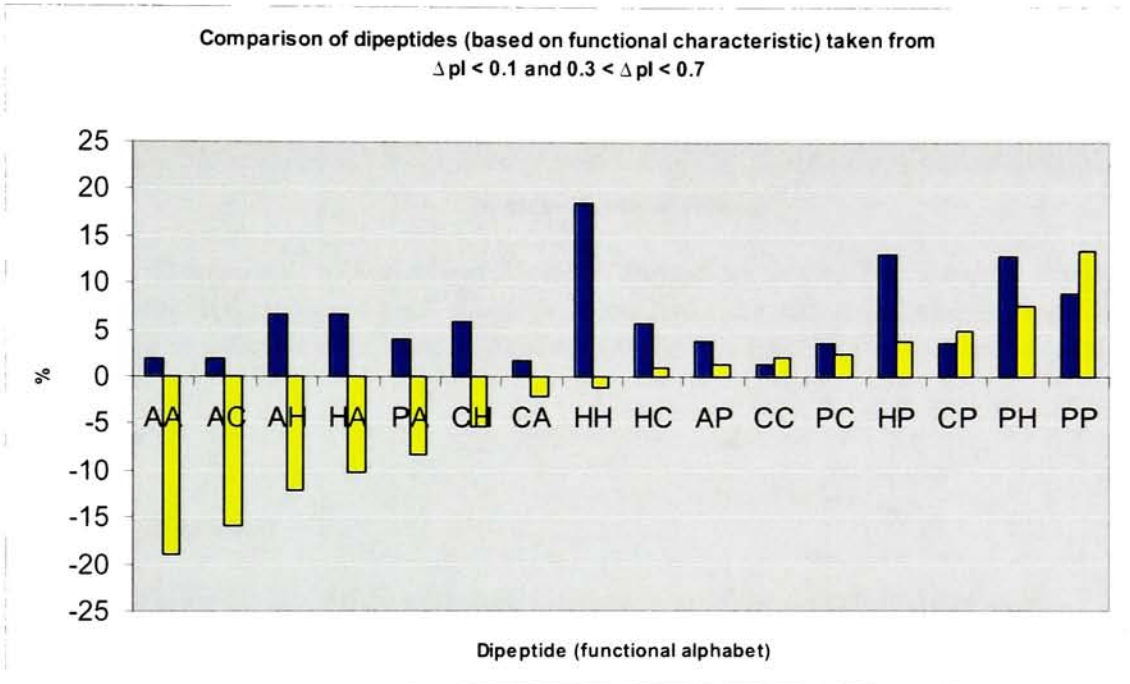


**Figure 21.** Frequencies of Functional Alphabet Dipeptides in Two ΔpI Subsets. Shown in blue are the frequencies of each dipeptide in the $\Delta pI < 0.1$ subset and shown in yellow is difference in frequency for each didpeptide between the $\Delta pI < 0.1$ subset and the $0.3 < \Delta pI < 0.7$ subset.

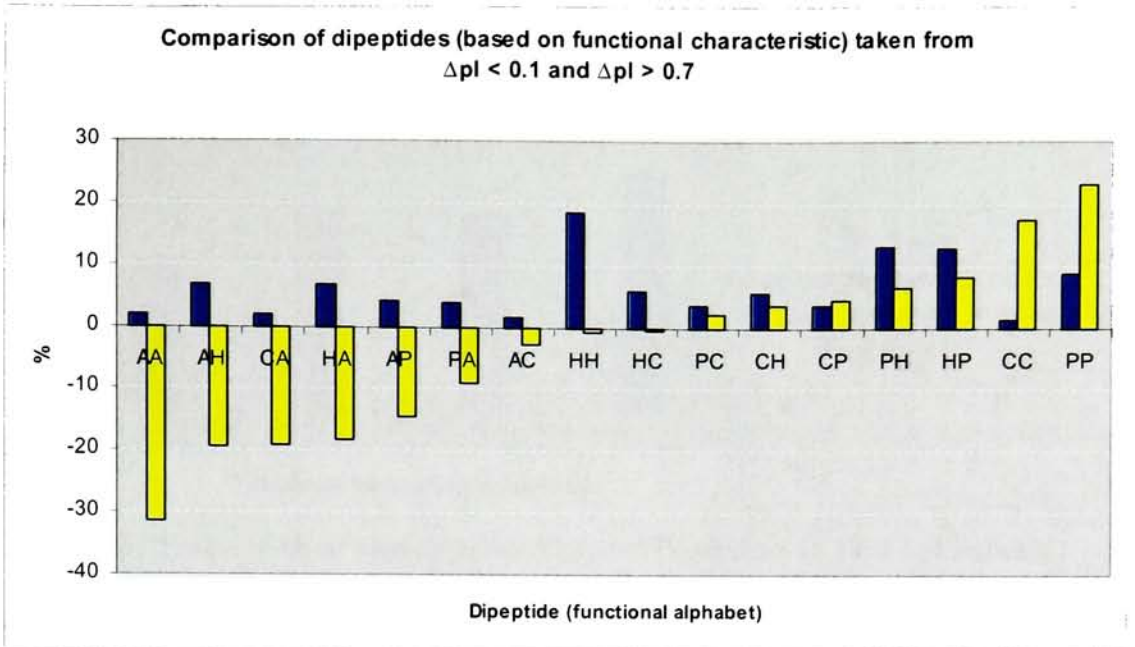


**Figure 22.** Frequencies of Functional Alphabet Dipeptides in Two ΔpI Subsets. Shown in blue are the frequencies of each dipeptide in the ΔpI < 0.1 subset and shown in yellow is difference in frequency for each didpeptide between the ΔpI < 0.1 subset and the ΔpI > 0.7 subset.

*-Hydrophobic*

Using the Hydrophobic alphabet, a comparison of the dipeptide frequencies between the ΔpI < 0.1 subset and the 0.3 < ΔpI < 0.7 subset is shown in Figure 23 as well as the Delta % values for each dipeptide. The same comparison is shown between the ΔpI < 0.1 subset and the ΔpI > 0.7 subset in Figure 24.

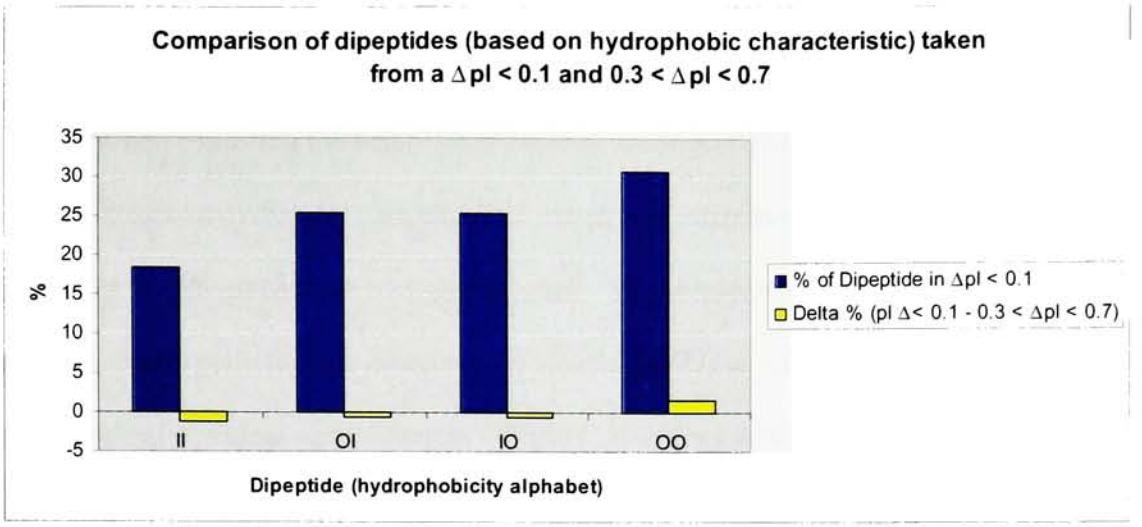


**Figure 23.** Frequencies of Hydrophobic Alphabet Dipeptides in Two ΔpI Subsets. Shown in blue are the frequencies of each dipeptide in the ΔpI < 0.1 subset and shown in yellow is difference in frequency for each didpeptide between the ΔpI < 0.1 subset and the 0.3 < ΔpI < 0.7 subset.

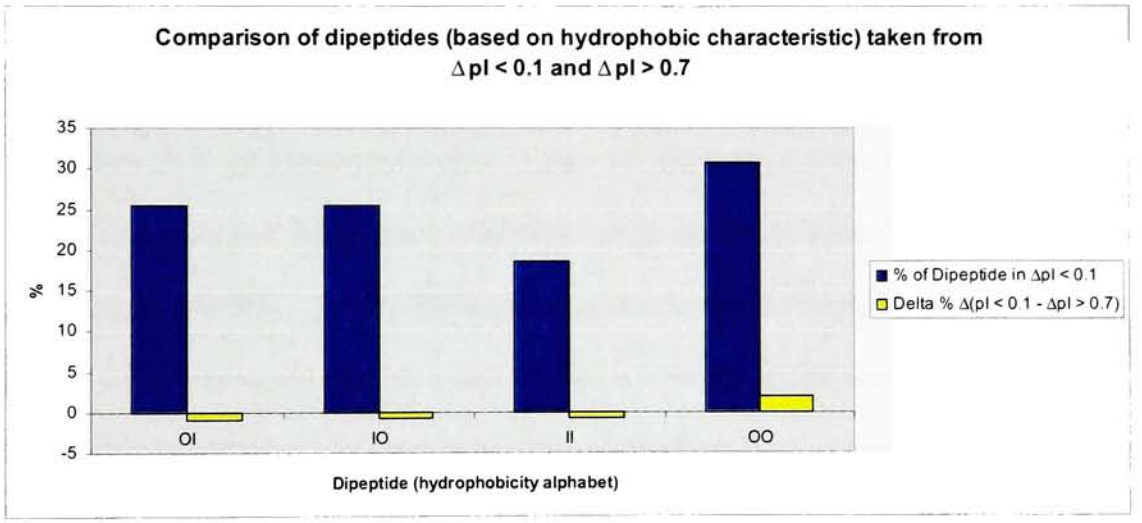


**Figure 24.** Frequencies of Hydrophobic Alphabet Dipeptides in Two ΔpI Subsets. Shown in blue are the frequencies of each dipeptide in the ΔpI < 0.1 subset and shown in yellow is difference in frequency for each didpeptide between the ΔpI < 0.1 subset and the ΔpI > 0.7 subset.

# Discussion

When exploring the behavior of proteins undergoing isoelectric focusing, there exists a discrepancy between predicted pI values and experimentally determined pI values for a high percentage of those proteins. This comparison of pI values was performed using predictions based on our algorithm ***(11)*** or similar algorithms ***(19)*** and experimental pI values determined in different laboratory settings ***(14-18)***. The size and regular occurrence of these differences justified a close study of the protein sequences in an effort to identify underlying patterns that could contribute to these differences. The question now lay in whether there was enough information in the results that were extracted to be able to more accurately predict pI values using the information obtained.

The first key element was having a reliable data set that was both uniform and robust enough to give meaningful data. A data set that is too diverse would lead to complications such as the question of how to handle post-translational modifications in predicting pI and MW. Simply finding the frequencies of all dipeptides in all known protein sequences would provide a data set that is certainly robust enough. Unfortunately, the robustness would be offset by the high level of noise in the data due to the fact that different organisms have different post-translational modifications. A data set that is too small would not have enough dipeptide information to make sure that the dipeptides that occur in the lowest frequencies are still seen in sufficient abundance to maintain their statistical validity. To overcome both of these hurdles, the search space was limited only to proteins in *E. coli* since it displays very few post-translational modifications and has a proteome that has been sufficiently documented to do a case study.

In keeping with the theme of having a data set with as little noise as possible, yet still retaining as much robustness as possible it was decided that even though well structured 2DE data existed from 5 different groups ***(14-18)***, it was probably best to limit the usage of this data to one or two of these groups ***(17 and 18)***. Both the Yan et al. ***(18)*** and Tonella et al. ***(19)*** groups performed large scale 2DE studies on the *E. coli* proteome. The Tonella ***(19)*** group boasted over 70% of the *E. coli* proteome being covered in their data. Since none of the groups used the same 2DE conditions it was decided that the data from the Tonella ***(19)*** group would be the only data used. The primary justification was to ensure that the experimental pI and MW values were gained using the same conditions. This in turn would reduce as much noise as possible. In addition, the fact that their data covered over 70% of the *E. coli* genome held promise for this study.

Once the entire data set was selected, another decision had to be made about how to separate the data so that clear lines could be seen between proteins that had very small ΔpI values and proteins that had greater ΔpI values. Doing so would make it possible to see if significant sequence differences (at the dipeptide level) between ΔpI subsets existed. It was necessary to break the data set into a small number of ΔpI subsets. These arbitrary ΔpI cut-off ranges ($\Delta pI < 0.1$; $0.3 < \Delta pI < 0.7$; $\Delta pI > 0.7$) were chosen in order to separate the data into distinct sets of similar size that could be compared with each other.

There was difficulty in deciding how to separate the entire dataset into these three subsets. One possible approach was to separate the dataset into many smaller sized subsets based on a larger number of ΔpI ranges. On one hand doing this might provide an answer that gives a scaled description of what is happening at each small ΔpI range

relative to adjacent ΔpI ranges. On the other hand by doing it this way, there is a loss of information at the sequence level due to the smaller number of sequences that would be found in each data set. This, in turn, would threaten the reliability of our findings. Therefore, the dataset had to be separated into subsets of sufficient robustness. The ΔpI < 0.1 subset consists of 60 proteins which comprise 22472 total amino acids or 22412 total dipeptides. The 0.3 < ΔpI < 0.7 subset consists of 58 proteins which comprise 17906 total amino acids or 17848 total dipeptides. The ΔpI > 0.7 subset consists of 50 proteins which comprise 15581 total amino acids or 15531 total dipeptides. More information about each individual protein in these ΔpI subsets, including ΔpI, a description and SWISS-2DPAGE Accession Number, can be seen in Appendix A.

The analytical process is best viewed as a pipeline as seen in Figure 2 in the Methods section. We began our analysis with the most simple method (naïve approach), work their way to more complicated methods (alphabets approach), and end with the most complicated methods (dipeptides using alphabets approach). Along this path, the relevance of the data also becomes more complicated, but more interesting at the same time (with a few exceptions).

The naïve approach to handling the data set did not provide any meaningful results. It was quickly apparent that individual amino acid frequencies in a given set of protein sequences did not vary among the three data subsets. In the end, no amino acid frequency characteristics using simply the naïve approach were found to be significantly different between the three ΔpI subsets. This can be seen in Figures 3 and 4 when comparing the ΔpI < 0.1 subset with the 0.3 < ΔpI < 0.7 subset and the ΔpI < 0.1 subset with the ΔpI > 0.7 subset, respectively. No significant difference between the blue and

yellow frequencies can be seen for any individual amino acid; the values are also nearly identical when Figure 3 and Figure 4 are compared, as well. The lack of a correlation between ΔpI values and the frequency of these individual amino acids showed us that we needed to consider the problem in more depth – more than one amino acid at a time.

To simplify the analysis, the number of variables was reduced by using the four alphabets described in Table 2 at the next stage in the pipeline. Again, the results did not reveal any significant trends that could affect the way that pI is calculated. Figures 5 and 6 (Charge alphabet comparisons), Figures 7 and 8 (Chemical alphabet comparisons), Figures 9 and 10 (Functional alphabet comparisons), and Figures 11 and 12 (Hydrophobic alphabet comparisons) show very similar results to that of the naïve approach in Figures 3 and 4. There is no trend of increase or decrease in ΔpI for any particular amino acid when moving between the three datasets.

It was expected that more meaningful results would be obtained by analysis of the dipeptide frequencies. All previous pI prediction algorithms ***(2-8)***, including ours ***(11)*** treat the $pK_A$ for each amino acid independently, regardless of its near or distant neighbors. At this point it is instructive to consider the experimental conditions normally employed for isoelectric focusing (IEF). The biological function of proteins requires that they maintain their three dimensional structure intact. However, for IEF, we are interested only in separating the proteins, not observing their biological function. To assure the best separation, reagents such as urea and detergents are added prior to IEF to disrupt any secondary, tertiary or quaternary aspects of protein structure. In these fully denatured proteins, the only significant interactions are expected to occur between amino acids side chains that are close to each other in the primary sequence. Thus a

consideration of the effect of neighboring amino acids on their respective side chain $pK_A$ values may prove valuable.

With respect to each alphabet that was used the discussion will advance from the least significant alphabet dipeptide results to the most significant alphabet dipeptide results. However, the analysis using the normal amino acid alphabet will be discussed first. At first glance Figures 13 and 14 show some very promising results. The Delta % value represents the change in frequency from one ΔpI subset being compared to the next ΔpI subset. Therefore, Delta % values that are in the 300 and 400 ranges would seem very significant. The problem was that most of the dipeptides that fell into these extreme ranges were dipeptides that whose overall frequency was vanishingly small. A dipeptide that occurs only once in one ΔpI subset and multiple times in another ΔpI subset is going to have a very high Delta % value. It would not be wise to rely on such dipeptide frequencies to redesign of a pI prediction algorithm. To negotiate through all of the 400 dipeptides in the normal alphabet, the same analysis was run with a threshold frequency occurrence for dipeptides 0.1%. In other words, if a dipeptide did not occur in at least 0.1% of the time (or at least 22 times in the $\Delta pI < 0.1$ dataset, which contained 22412 amino acids) it was not used for analysis. The results of this can be seen in Figures 15 and 16. There still exist extreme outliers that have Delta % values in the 100 range which will later be reanalyzed by comparison with some of the alphabet dipeptide analyses.

The alphabet that showed the least interesting results when using a dipeptide approach was the hydrophobic alphabet. Comparisons of the ΔpI subsets using the hydrophobic alphabet can be seen in Figures 23 and 24. Delta % is shown in the yellow

bars and is very negligible in each of the 4 dipeptides (no more than a Delta % value of 1.85 was seen in any of the 4 dipeptides).

The charge alphabet showed slightly more significant results for dipeptide anaylsis. Delta % values reached into the 30+ range for some dipeptides. The AA dipeptide (negatively charged amino acid followed by negatively charged amino acid; see Table 2 for definitions of all the alphabet codes) had a Delta % of -31.3% going from the $\Delta pI < 0.1$ subset to the $\Delta pI > 0.7$ subset (Figure 18). The Delta % for the AA dipeptide is also large (-18.9%) in the other comparison of the $\Delta pI < 0.1$ subset and the $0.3 < \Delta pI < 0.7$ subset (Figure 17). However, the frequency of occurrence of this AA dipeptide (as shown in the blue bars) is very low in all three $\Delta pI$ subsets. What we would like to see is a large Delta % value accompanied with a large frequency of occurrence for a particular dipeptide. This was not apparent in any of the dipeptides using the Charge alphabet.

Staying with the theme that the most significant results will combine large Delta % value along with a large frequency of occurrence value for dipeptides, the Functional alphabet is considered next. Figures 21 and 22 representing the analysis using the Functional alphabet show a collection of dipeptides that have both significantly large Delta % values and significantly large frequencies of occurrence: AA, AH, HA, HP, CP, PH, PP.

It was important to refer back to the analysis that was done using dipeptides based on the complete amino acid alphabet. Figures 15 and 16 point out a few extreme dipeptide outliers: KY, YS (Figure 15) and EE, NN, YT (Figure 16). Converting these dipeptides to the Functional alphabet gives the dipeptides: CP, PP and AA, PP, PP

respectively. These three different dipeptides all map back to extreme outliers from the analysis done using the Functional alphabet (Figures 21 and 22).

Using the Chemical alphabet with dipeptides created data that was sufficiently large that it was not possible to display all the possible dipeptide combinations in Figures 19 and 20. Instead only the density values were chosen to display. Particular outlier dipeptides are labeled on the top of each column with their respective Delta % values.

The significance of these findings is that it may lead to a more accurate calculation of pI than currently existing methods ***(11, 19)***. These data clearly support the idea that the $pK_A$ value for an amino acid side chain, even when the protein is fully denatured, depends on the microenvironment created by the nearest neighbors of that amino acid. Using the extreme outlier dipeptides that have been identified from this study of 180 annotated *E. coli* proteins, it may be possible to adjust the algorithms for calculating pI values. Our algorithm for calculating pI from amino acid sequence ***(11)*** could be modified to include the effects of adjacent amino acids on the $pK_A$ values used in the calculations. This will be an empirical process whereby the $pK_A$ values used in the algorithm will be modified fractionally to see which changes lead to a better correlation between actual and predicted pI values for the two outlier data sets ($0.3 < \Delta pI < 0.7$; and $\Delta pI > 0.7$).

If the improvement of the accuracy of the pI calculation proves to be worthy there any many future advancements that could be made. The first could be to build a larger data set to work with and rerun the analysis to compare to the data shown here. Beyond the scope of the *E. coli* proteome, further data that are available at the ExPASy Server's SWISS-2DPAGE database could be used to perform similar analyses on many other

microbial proteomes. Another step would be to port the analysis over to lower eukaryotic proteomes that contain much more post-translational modifications. A lot would have to be done in terms of predicting or categorizing these post-translational modifications but in doing so it may lead to an even more powerful approach to better predicting pI in higher organisms as well.

# Conclusions

A dataset of *E. coli* proteins was collected and formatted to study the discrepancy that exists between experimental isoelectric point and predicted isoelectric point (ΔpI). This dataset was then split into three parts depending on the magnitude of ΔpI for each protein. Several, multi-layered, sequential approaches were taken in reformatting the protein sequence data in an attempt to get a better understanding of what might be causing the varying ΔpI. Each of these stages represented a different part of a pipeline where the data were analyzed by comparing each of the three ΔpI subsets to one another. The pipeline consisted of a naïve approach (considering individual amino acid frequencies), followed by the application four different alphabets to represent sequences in a simpler way by grouping similar amino acids based on their charge, functional, chemical, and hydrophobic properties . The final step in the pipeline involved investigating the dipeptides of all of these sequences using both the 20 amino acid alphabet and the simplified groupings. The alphabet dipeptide approach yielded the most meaningful results showing that certain dipeptide sequences occur in greatly different frequency between proteins in the different ΔpI subsets.

Future studies will attempt to show that the results of these dipeptide findings better can be used to better predict pI. This will involve modification of our existing pI prediction algorithm to include the affect of adjacent amino acids in side chain $pK_A$ values. Using a short list of only the most extreme cases where a dipeptide showed greatly different ΔpI from one subset to the next should result in a pI prediction value that is more accurate. Once the pI prediction is improved the next step would be to concentrate on post-translational modifications and how pI prediction can be altered by

them. In addition, similar analyses will be extended to other prokaryotic organisms, and eventually to eukaryotic organisms.

# References


1. Fey, S.J. and Larsen, P.M. "2D or not 2D." *Current Opinion in Chemical Biology* 5: 26-33(2001).
2. Cargile, B.J., Talley, D.L., Stephenson, J.L. "Immobilized pH gradients as a first dimension in shotgun proteomics and analysis of the accuracy of pI predictability of peptides." *Electrophoresis* 25: 936-945(2004).
3. Patrickios, C.S., Yamasaki, E.N. "Polypeptide Amino Acid Composition and Isoelectric point." *Analytical Biochemistry* 231: 82-91(1995).
4. Ribeiro, J.M. and Sillero, A. "An algorithm for the computer calculation of the coefficients of a polynomial that allows determination of isoelectric points of proteins and other macromolecules." *Comput. Biol. Med.* 20: 235-242(1990).
5. Ribeiro, J.M. and Sillero, A. "A program to calculate the isoelectric point of macromolecules." *Comput. Biol. Med.* 21: 131-141(1991).
6. Ribeiro, J.M., Ruiz A., Sillero, M.A., Sillero, A. "Theoretical isoelectric points and electric charges of mutated human hemoglobin subunits." *Clin. Chim. Acta.* 190: 189-197(1990).
7. Sillero A., Ribeiro, J.M. "Isoelectric points of proteins: theoretical determination." *Analytical Biochemistry* 179: 319-325(1989).
8. Righetti, P.G., Caravaggio, T. "Isoelectric points and molecular weights of proteins." *Journal of Chromatrography* 127: 1-28(1976).
9. Berg J, Tymoczko J, Stryer L. *Biochemistry.* New York: W. H. Freeman and Co; 2002.
10. "Image: Amino acids 2.png" *Wikipedia: The Free Encyclopedia.* Found at http://upload.wikimedia.org/wikipedia/en/c/c5/Amino_acids_2.png
11. Zapoticnyj J., Conte M.C., Craig P.A. "Simulation of 2D Gel Electrophoresis." http://www.rit.edu/~pac8612/2DE/2D_Sim.html
12. Bioperl. Found at http://bioperl.org
13. "SWISS-2DPAGE Two-dimensional polyacrylamide gel electrophoresis database." Found at http://us.expasy.org/ch2d/
14. Phillips T.A., Bloch P.L., Neidhardt F.C. "Protein identifications on O'Farrell two-dimensional gels: locations of 55 additional Escherichia coli proteins." *J. Bacteriol.* 144:1024-1033(1980).
15. Pasquali C., Frutiger S., Wilkins M.R., Hughes G.J., Appel R.D., Bairoch A., Schaller D., Sanchez J.-C., Hochstrasser D.F. "Two-dimensional gel electrophoresis of Escherichia coli homogenates: the Escherichia coli SWISS-2DPAGE database." *Electrophoresis* 17:547-555(1996).
16. Vanbogelen R.A., Abshire K.Z., Pertsemlidis A., Clark R.L., Neidhardt F.C.; "Gene-protein database of Escherichia coli K-12, edition 6" (In) Neidhardt et al. (eds.) *Escherichia coli and Salmonella: Cellular and Molecular Biology (2nd ed.)*, pp.2067-2117, ASM Press, Washington DC (1996).
17. Tonella L., Hoogland C., Binz P.-A., Appel R.D., Hochstrasser D.F., Sanchez J.-C. "New perspectives in the Escherichia coli proteome investigation." *Proteomics* 1:409-423(2001).

18. Yan J.X., Devenish A.T., Wait R., Stone T., Lewis S., Fowler S. "Fluorescence 2-D difference gel electrophoresis and mass spectrometry based proteomic analysis of E. coli." *Proteomics* 2:1682-1698(2002).
19. "Compute pI/Mx for Swiss-Prot/TrEMBL entries or a user-entered sequence." Found at http://us.expasy.org/tools/pi_tool.html
20. Bjellqvist, B., Hughes, G., Pasquali, C., Paquet, N., Ravier, F., Sanchez, J.-C., et al. (1993) "The focusing positions of polypeptides in immobilized pH gradients can be predicted from their amino acid sequences." *Electrophoresis* 14:1023-1031.
21. "Get protein list for a reference map." Found at http://www.expasy.org/cgi-bin/get-ch2d-table.pl

# Appendix A

The sequences included in each of the three ΔpI subsets that were used. The ΔpI subsets are ΔpI values less than 0.1 ("ΔpI < 0.1"), ΔpI values greater than 0.3, but less than 0.7 ("0.3 < ΔpI < 0.7"), and ΔpI values greater than 0.7 ("ΔpI > 0.7") displayed below in that order. Included is the gene name, protein description, SWISS-2DPAGE Accession Number and ΔpI (experimental pI – predicted pI).

## pI < 0.1

| Gene Name | Protein Description | SWISS-2DPAGE Access # | Δpl |
|---|---|---|---|
| ACCB | Biotin carboxyl carrier protein of acetyl-CoA carboxylase (BCCP) | P02905 | 0.1 |
| ACEA | Isocitrate lyase (EC 4.1.3.1) (Isocitrase) (Isocitratase) (ICL) | P05313 | -0.04 |
| ACNB | Aconitate hydratase 2 (EC 4.2.1.3) (Citrate hydro-lyase 2) (Aconitase 2) | P36683 | -0.07 |
| AHPC | Alkyl hydroperoxide reductase subunit C (EC 1.6.4.-) (Alkyl hydroperoxide reductase protein C22) | P26427 | 0.01 |
| AMPC | Beta-lactamase (EC 3.5.2.6) (Cephalosporinase) | P00811 | 0.08 |
| ARGF | Ornithine carbamoyltransferase chain F (EC 2.1.3.3) (OTCase-2) | P06960 | 0 |
| ARGG | Argininosuccinate synthase (EC 6.3.4.5) (Citrulline--aspartate ligase) | P22767 | -0.03 |
| AROD | 3-dehydroquinate dehydratase (EC 4.2.1.10) (3-dehydroquinase) (Type I DHQase) | P05194 | 0.05 |
| ATPA | ATP synthase alpha chain (EC 3.6.3.14) | P00822 | -0.01 |
| ATPD | ATP synthase beta chain (EC 3.6.3.14) | P00824 | 0.02 |
| ATPD | ATP synthase beta chain (EC 3.6.3.14) | P00824 | 0.03 |
| CHEY | Chemotaxis protein cheY | P06143 | 0.05 |
| CLPB | ClpB protein (Heat shock protein F84.1) | P03815 | -0.02 |
| CYSM | Cysteine synthase B (EC 2.5.1.47) (O-acetylserine sulfhydrylase B) | P16703 | -0.05 |
| DEOC | Deoxyribose-phosphate aldolase (EC 4.1.2.4) (Phosphodeoxyriboaldolase) (Deoxyriboaldolase) (DERA) | P00882 | -0.1 |
| DNAK | Chaperone protein dnaK (Heat shock protein 70) (Heat shock 70 kDa protein) (HSP70) | P04475 | 0.08 |
| DNAK | Chaperone protein dnaK (Heat shock protein 70) (Heat shock 70 kDa protein) (HSP70) | P04475 | 0.1 |
| DPS | DNA protection during starvation protein | P27430 | -0.05 |
| ENO | Enolase (EC 4.2.1.11) (2-phosphoglycerate dehydratase) (2-phospho-D-glycerate hydro-lyase) | P08324 | 0.05 |
| FABD | Malonyl CoA-acyl carrier protein transacylase (EC 2.3.1.39) (MCT) | P25715 | 0.04 |
| FLIC | Flagellin | P04949 | 0.07 |
| FUSA | Elongation factor G (EF-G) | P02996 | -0.1 |
| FUSA | Elongation factor G (EF-G) | P02996 | -0.08 |
| GALM | Aldose 1-epimerase (EC 5.1.3.3) (Mutarotase) | P40681 | -0.01 |
| GLNK | Nitrogen regulatory protein P-II 2 | P38504 | -0.1 |
| GPMI | 2,3-bisphosphoglycerate-independent phosphoglycerate mutase (EC 5.4.2.1) (Phosphoglyceromutase) | P37689 | -0.04 |

| | | | |
|---|---|---|---|
| GROL | 60 kDa chaperonin (Protein Cpn60) (groEL protein) | P06139 | -0.02 |
| GROL | 60 kDa chaperonin (Protein Cpn60) (groEL protein) | P06139 | 0.1 |
| GROS | 10 kDa chaperonin (Protein Cpn10) (groES protein) | P05380 | 0.03 |
| ICD | Isocitrate dehydrogenase [NADP] (EC 1.1.1.42) (Oxalosuccinate decarboxylase) | P08200 | -0.05 |
| KATG | Peroxidase/catalase HPI (EC 1.11.1.6) (Catalase-peroxidase) (Hydroperoxidase I) | P13029 | -0.03 |
| LIVJ | Leu/Ile/Val-binding protein (LIV-BP) | P02917 | 0.01 |
| LIVJ | Leu/Ile/Val-binding protein (LIV-BP) | P02917 | 0.07 |
| LIVK | Leucine-specific binding protein (LS-BP) (L-BP) | P04816 | -0.06 |
| LUXS | S-ribosylhomocysteinase (EC 3.13.1.-) (Autoinducer-2 production protein luxS) (AI-2 synthesis protein) | P45578 | -0.08 |
| MAP | Methionine aminopeptidase (EC 3.4.11.18) (MAP) (Peptidase M) | P07906 | -0.07 |
| METK | S-adenosylmethionine synthetase (EC 2.5.1.6) (Methionine adenosyltransferase) | P04384 | -0.02 |
| MIND | Septum site-determining protein minD (Cell division inhibitor minD) | P18197 | 0.01 |
| NADE | NH(3)-dependent NAD(+) synthetase (EC 6.3.1.5) (Nitrogen-regulatory protein) | P18843 | -0.06 |
| PGK | Phosphoglycerate kinase (EC 2.7.2.3) | P11665 | 0.03 |
| PNP | Polyribonucleotide nucleotidyltransferase (EC 2.7.7.8) (Polynucleotide phosphorylase) (PNPase) | P05055 | -0.02 |
| PPA | Inorganic pyrophosphatase (EC 3.6.1.1) (Pyrophosphate phospho-hydrolase) (PPase) | P17288 | -0.01 |
| PURK | Phosphoribosylaminoimidazole carboxylase ATPase subunit (EC 4.1.1.21) (AIR carboxylase) (AIRC) | P09029 | -0.1 |
| RIBH | 6,7-dimethyl-8-ribityllumazine synthase (EC 2.5.1.9) (DMRL synthase) | P61714 | -0.02 |
| RPLL | 50S ribosomal protein L7/L12 (L8) | P02392 | 0.09 |
| RPOA | DNA-directed RNA polymerase alpha chain (EC 2.7.7.6) (RNAP alpha subunit) | P00574 | 0.02 |
| RPSA | 30S ribosomal protein S1 | P02349 | 0.08 |
| RPSA | 30S ribosomal protein S1 | P02349 | 0.08 |
| SERC | Phosphoserine aminotransferase (EC 2.6.1.52) (PSAT) | P23721 | 0 |
| SSB | Single-strand binding protein (SSB) (Helix-destabilizing protein) | P02339 | -0.05 |
| TALB | Transaldolase B (EC 2.2.1.2) | P30148 | -0.05 |
| TIG | Trigger factor (TF) | P22257 | 0.02 |
| TIG | Trigger factor (TF) | P22257 | 0.01 |
| TRPA | Tryptophan synthase alpha chain (EC 4.2.1.20) | P00928 | -0.02 |
| TSF | Elongation factor Ts (EF-Ts) | P02997 | -0.05 |
| TUFA | Elongation factor Tu (EF-Tu) (P-43) | P02990 | 0.01 |
| USPA | Universal stress protein A | P28242 | 0.04 |
| YCII | Protein ycil | P31070 | 0.01 |
| YFID | Protein yfiD | P33633 | -0.04 |
| YJDC | Putative HTH-type transcriptional regulator yjdC | P36656 | 0.02 |

## 0.3 < pI < 0.7

| Gene Name | Protein Description | SWISS-2DPAGE Access # | ΔpI |
|---|---|---|---|
| ACCB | Biotin carboxyl carrier protein of acetyl-CoA carboxylase (BCCP) | P02905 | 0.37 |
| ACKA | Acetate kinase (EC 2.7.2.1) (Acetokinase) | P15046 | -0.46 |
| ADK | Adenylate kinase (EC 2.7.4.3) (ATP-AMP transphosphorylase) | P05082 | -0.67 |

| | | | |
|---|---|---|---|
| ADK | Adenylate kinase (EC 2.7.4.3) (ATP-AMP transphosphorylase) | P05082 | -0.68 |
| AHPF | Alkyl hydroperoxide reductase subunit F (EC 1.6.4.-) (Alkyl hydroperoxide reductase F52A protein) | P35340 | -0.41 |
| ALDA | Aldehyde dehydrogenase A (EC 1.2.1.22) (Lactaldehyde dehydrogenase) | P25553 | 0.42 |
| ALDA | Aldehyde dehydrogenase A (EC 1.2.1.22) (Lactaldehyde dehydrogenase) | P25553 | 0.32 |
| ARGT | Lysine-arginine-ornithine-binding periplasmic protein (LAO-binding protein) | P09551 | -0.48 |
| ATPC | ATP synthase epsilon chain (EC 3.6.3.14) (ATP synthase F1 sector epsilon subunit) | P00832 | -0.33 |
| ATPD | ATP synthase beta chain (EC 3.6.3.14) | P00824 | 0.02 |
| CLPS | ATP-dependent Clp protease adaptor protein clpS | P75832 | 0.44 |
| DAPB | Dihydrodipicolinate reductase (EC 1.3.1.26) (DHPR) | P04036 | -0.54 |
| DNAK | Chaperone protein dnaK (Heat shock protein 70) (Heat shock 70 kDa protein) (HSP70) | P04475 | -0.37 |
| ENO | Enolase (EC 4.2.1.11) (2-phosphoglycerate dehydratase) (2-phospho-D-glycerate hydro-lyase) | P08324 | -0.67 |
| ENO | Enolase (EC 4.2.1.11) (2-phosphoglycerate dehydratase) (2-phospho-D-glycerate hydro-lyase) | P08324 | -0.41 |
| FABI | Enoyl-[acyl-carrier-protein] reductase [NADH] (EC 1.3.1.9) (NADH-dependent enoyl-ACP reductase) | P29132 | -0.45 |
| FLIC | Flagellin | P04949 | 0.51 |
| GLNA | Glutamine synthetase (EC 6.3.1.2) (Glutamate--ammonia ligase) | P06711 | -0.38 |
| GPT | Xanthine-guanine phosphoribosyltransferase (EC 2.4.2.22) (XGPRT) | P00501 | -0.51 |
| GST | Glutathione S-transferase (EC 2.5.1.18) | P39100 | -0.37 |
| HISJ | Histidine-binding periplasmic protein (HBP) | P39182 | -0.52 |
| HISJ | Histidine-binding periplasmic protein (HBP) | P39182 | -0.37 |
| ILVH | Acetolactate synthase isozyme III small subunit (EC 2.2.1.6) (AHAS-III) (Acetohydroxy-acid synthase III small subunit) (ALS-III) | P00894 | -0.54 |
| KDSA | 2-dehydro-3-deoxyphosphooctonate aldolase (EC 2.5.1.55) (Phospho-2-dehydro-3-deoxyoctonate aldolase) (3-deoxy-D-manno-octulosonic acid 8-phosphate synthetase) (KDO-8-phosphate synthetase) (KDO 8-P synthase) (KDOPS) | P17579 | -0.62 |
| LIVJ | Leu/Ile/Val-binding protein (LIV-BP) | P02917 | -0.45 |
| LIVJ | Leu/Ile/Val-binding protein (LIV-BP) | P02917 | -0.57 |
| LIVK | Leucine-specific binding protein (LS-BP) (L-BP) | P04816 | -0.54 |
| LIVK | Leucine-specific binding protein (LS-BP) (L-BP) | P04816 | -0.37 |
| MALE | Maltose-binding periplasmic protein (Maltodextrin-binding protein) (MMBP) | P02928 | -0.41 |
| MALE | Maltose-binding periplasmic protein (Maltodextrin-binding protein) (MMBP) | P02928 | -0.62 |
| MDH | Malate dehydrogenase (EC 1.1.1.37) | P61889 | -0.47 |
| MDOG | Glucans biosynthesis protein G | P33136 | -0.64 |
| MGLB | D-galactose-binding periplasmic protein (GBP) (D-galactose/ D-glucose binding protein) (GGBP) | P02927 | -0.67 |
| NDK | Nucleoside diphosphate kinase (EC 2.7.4.6) (NDK) (NDP kinase) (Nucleoside-2-P kinase) | P24233 | -0.56 |
| NFNB | Oxygen-insensitive NAD(P)H nitroreductase (EC 1.-.-.-) (FMN-dependent nitroreductase) (Dihydropteridine reductase) (EC 1.5.1.34) | P38489 | -0.57 |
| NFNB | Oxygen-insensitive NAD(P)H nitroreductase (EC 1.-.-.-) (FMN-dependent nitroreductase) (Dihydropteridine reductase) (EC 1.5.1.34) | P38489 | -0.65 |
| NUSG | Transcription antitermination protein nusG | P16921 | -0.35 |
| PHES | Phenylalanyl-tRNA synthetase alpha chain (EC 6.1.1.20) (Phenylalanine--tRNA ligase alpha chain) (PheRS) | P08312 | -0.36 |
| POTD | Spermidine/putrescine-binding periplasmic protein (SPBP) | P23861 | -0.32 |
| PPIA | Peptidyl-prolyl cis-trans isomerase A (EC 5.2.1.8) (PPIase A) (Rotamase A) (Cyclophilin A) | P20752 | -0.59 |

| | | | |
|---|---|---|---|
| PYRI | Aspartate carbamoyltransferase regulatory chain | P00478 | -0.48 |
| RTCB | Protein rtcB | P46850 | -0.6 |
| RTCB | Protein rtcB | P46850 | -0.57 |
| SBP | Sulfate-binding protein (Sulfate starvation-induced protein 2) (SSI2) | P06997 | -0.59 |
| SERC | Phosphoserine aminotransferase (EC 2.6.1.52) (PSAT) | P23721 | -0.45 |
| SODB | Superoxide dismutase [Fe] (EC 1.15.1.1) | P09157 | -0.41 |
| SSPB | Stringent starvation protein B | P25663 | 0.64 |
| TOLB | TolB protein | P19935 | -0.57 |
| TRXA | Thioredoxin 1 (TRX1) (TRX) | P00274 | 0.3 |
| UDP | Uridine phosphorylase (EC 2.4.2.3) (UrdPase) (UPase) | P12758 | -0.47 |
| UDP | Uridine phosphorylase (EC 2.4.2.3) (UrdPase) (UPase) | P12758 | -0.42 |
| YAET | Unknown protein from 2D-page spots M62/M63/O3/O9/T35 | P39170 | 0.39 |
| YCEI | Protein yceI | P37904 | -0.56 |
| YCGK | Protein ycgK | P76002 | -0.57 |
| YGIN | Protein ygiN | P40718 | -0.38 |
| YHGI | Protein yhgI | P46847 | 0.42 |
| ZNUA | High-affinity zinc uptake system protein znuA | P39172 | -0.3 |
| ZNUA | High-affinity zinc uptake system protein znuA | P39172 | -0.37 |

# pI > 0.7

| Gene Name | Protein Description | SWISS-2DPAGE Access # | ΔpI |
|---|---|---|---|
| ARGD | Acetylornithine/succinyldiaminopimelate aminotransferase (EC 2.6.1.11) (EC 2.6.1.17) (ACOAT) (Succinyldiaminopimelate transferase) (DapATase) | P18335 | -0.94 |
| ARTI | Arginine-binding periplasmic protein 1 | P30859 | -0.91 |
| CYSK | Cysteine synthase A (EC 2.5.1.47) (O-acetylserine sulfhydrylase A) (O-acetylserine (Thiol)-lyase A) (CSase A) (Sulfate starvation-induced protein 5) (SSI5) | P11096 | -1.09 |
| CYSK | Cysteine synthase A (EC 2.5.1.47) (O-acetylserine sulfhydrylase A) (O-acetylserine (Thiol)-lyase A) (CSase A) (Sulfate starvation-induced protein 5) (SSI5) | P11096 | -1.07 |
| CYSK | Cysteine synthase A (EC 2.5.1.47) (O-acetylserine sulfhydrylase A) (O-acetylserine (Thiol)-lyase A) (CSase A) (Sulfate starvation-induced protein 5) (SSI5) | P11096 | -0.92 |
| CYSP | Thiosulfate-binding protein | P16700 | -1.75 |
| DEGP | Protease do (EC 3.4.21.-) | P09376 | -0.9 |
| DPPA | Periplasmic dipeptide transport protein (Dipeptide-binding protein) (DBP) | P23847 | -0.85 |
| DPPA | Periplasmic dipeptide transport protein (Dipeptide-binding protein) (DBP) | P23847 | -0.75 |
| DPS | DNA protection during starvation protein | P27430 | -0.78 |
| ENO | Enolase (EC 4.2.1.11) (2-phosphoglycerate dehydratase) (2-phospho-D-glycerate hydro-lyase) | P08324 | 1.54 |
| ENO | Enolase (EC 4.2.1.11) (2-phosphoglycerate dehydratase) (2-phospho-D-glycerate hydro-lyase) | P08324 | 1.31 |
| FLIY | Cystine-binding periplasmic protein (CBP) (fliY protein) (Sulfate starvation-induced protein 7) (SSI7) | P39174 | -1.35 |
| GAPA | Glyceraldehyde-3-phosphate dehydrogenase A (EC 1.2.1.12) (GAPDH-A) | P06977 | -2.03 |
| GAPA | Glyceraldehyde-3-phosphate dehydrogenase A (EC 1.2.1.12) (GAPDH-A) | P06977 | -1.32 |
| GAPA | Glyceraldehyde-3-phosphate dehydrogenase A (EC 1.2.1.12) (GAPDH-A) | P06977 | -0.95 |
| GLNH | Glutamine-binding periplasmic protein (GlnBP) | P10344 | -1.64 |

| | | | |
|---|---|---|---|
| GLTI | Glutamate/aspartate periplasmic binding protein | P37902 | -1.01 |
| HDEB | Protein hdeB (10K-L protein) | P26605 | -0.94 |
| IVY | Inhibitor of vertebrate lysozyme | P45502 | -1.37 |
| LOLA | Outer-membrane lipoprotein carrier protein (P20) | P61316 | -1.19 |
| MANX | PTS system, mannose-specific IIAB component (EIIAB-Man) (Mannose-permease IIAB component) (Phosphotransferase enzyme II, AB component) (EC 2.7.1.69) (EIII-Man) | P08186 | -0.77 |
| MDH | Malate dehydrogenase (EC 1.1.1.37) | P61889 | 1.14 |
| MDOG | Glucans biosynthesis protein G | P33136 | 0.89 |
| MDOG | Glucans biosynthesis protein G | P33136 | -0.87 |
| MODA | Molybdate-binding periplasmic protein | P37329 | -1.71 |
| NFNB | Oxygen-insensitive NAD(P)H nitroreductase (EC 1.-.-.-) (FMN-dependent nitroreductase) (Dihydropteridine reductase) (EC 1.5.1.34) | P38489 | -1.03 |
| NFNB | Oxygen-insensitive NAD(P)H nitroreductase (EC 1.-.-.-) (FMN-dependent nitroreductase) (Dihydropteridine reductase) (EC 1.5.1.34) | P38489 | -0.8 |
| NLPD | Lipoprotein nlpD | P33648 | -0.87 |
| NUSG | Transcription antitermination protein nusG | P16921 | -1.31 |
| OMPA | Outer membrane protein A (Outer membrane protein II*) | P02934 | -1.09 |
| OPPA | Periplasmic oligopeptide-binding protein | P23843 | -1.26 |
| OPPA | Periplasmic oligopeptide-binding protein | P23843 | -0.74 |
| PANC | Pantoate--beta-alanine ligase (EC 6.3.2.1) (Pantothenate synthetase) (Pantoate activating enzyme) | P31663 | -0.93 |
| PSTS | Phosphate-binding periplasmic protein (PBP) | P06128 | -1.86 |
| PYRD | Dihydroorotate dehydrogenase (EC 1.3.3.1) (Dihydroorotate oxidase) (DHOdehase) (DHODase) (DHOD) | P05021 | -0.82 |
| RPLA | 50S ribosomal protein L1 | P02384 | -1.74 |
| RPLI | 50S ribosomal protein L9 | P02418 | -1.41 |
| RPLY | 50S ribosomal protein L25 | P02426 | 0.71 |
| RPME2 | 50S ribosomal protein L31 type B-1 | P71302 | -1.21 |
| SUCD | Succinyl-CoA synthetase alpha chain (EC 6.2.1.5) (SCS-alpha) | P07459 | -0.99 |
| SUCD | Succinyl-CoA synthetase alpha chain (EC 6.2.1.5) (SCS-alpha) | P07459 | -0.86 |
| SUHB | Inositol-1-monophosphatase (EC 3.1.3.25) (IMPase) (Inositol-1-phosphatase) (I-1-Pase) | P22783 | -1.17 |
| SUHB | Inositol-1-monophosphatase (EC 3.1.3.25) (IMPase) (Inositol-1-phosphatase) (I-1-Pase) | P22783 | -1.06 |
| TPIA | Triosephosphate isomerase (EC 5.3.1.1) (TIM) | P04790 | -0.88 |
| TRPB | Tryptophan synthase beta chain (EC 4.2.1.20) | P00932 | -0.97 |
| YGFZ | Unknown protein from 2D-page (Spot PR51) | P39179 | 0.91 |
| YGGX | UPF0269 protein yggX | P52065 | -0.78 |
| YLIB | Putative binding protein yliB | P75797 | -1.19 |
| YRBC | Protein yrbC | P45390 | 0.73 |

# Appendix B

The relevant Perl code that was used for any of the analysis described in the Materials and Methods section. Each Perl program is listed in alphabetical order by the name of the particular program. At the top of each program are comments that explain exactly what each program does, how to run the program (command line arguments), and the output of program.

**aacounts.pl**:

```
#!/bin/perl
use strict;
use Bio::Seq;
use Bio::SeqIO;

# Matthew Conte
#
# This script counts the number of each amino acids in a sequence from a FASTA
# file and determines the frequency of each.
# Output is to FASTAfilename.aacounts

# Usage: perl aacounts.pl file.FASTA file2.FASTA ...

#initialize count variables
my $n_A = 0; my $n_R = 0; my $n_N = 0; my $n_D = 0; my $n_C = 0; my Sn_E = 0;
my Sn_Q = 0; my $n_G = 0; my $n_H = 0; my $n_I = 0; my Sn_L = 0; my $n_K = 0;
my $n_M = 0; my $n_F = 0; my $n_P = 0; my Sn_S = 0; my Sn_T = 0; my Sn_W = 0;
my Sn_Y = 0; my $n_V = 0;
my $n_AA_Total = 0;

#initialize frequency variables
my $f_A = 0; my $f_R = 0; my $f_N = 0; my Sf_D = 0; my Sf_C = 0; my Sf_E = 0;
my Sf_Q = 0; my $f_G = 0; my $f_H = 0; my $f_I = 0; my $f_L = 0; my Sf_K = 0;
my $f_M = 0; my $f_F = 0; my $f_P = 0; my Sf_S = 0; my Sf_T = 0; my $f_W = 0;
my $f_Y = 0; my $f_V = 0;

foreach my $file(@ARGV) {
   print STDERR "Reading input file $file... \n";

   #open the FASTA file
```

```
my $FASTA_in = Bio::SeqIO->new(-file => $file);

#open the file's basename
$file =~ s/\.seq$//g;

#open the .aacounts file for writing
open AACOUNTS, ">$file.aacounts";

#for each sequence in the FASTA file..
while(my $FASTA_seq = $FASTA_in->next_seq()){

    #reset variables
    $n_A = 0; $n_R = 0; $n_N = 0; $n_D = 0; $n_C = 0; $n_E = 0; $n_Q = 0;
    $n_G = 0; $n_H = 0; $n_I = 0; $n_L = 0; $n_K = 0; $n_M = 0; $n_F = 0;
    $n_P = 0; $n_S = 0; $n_T = 0; $n_W = 0; $n_Y = 0; $n_V = 0;
    $n_AA_Total = 0;
    $f_A = 0; $f_R = 0; $f_N = 0; $f_D = 0; $f_C = 0; $f_E = 0; $f_Q = 0;
    $f_G = 0; $f_H = 0; $f_I = 0; $f_L = 0; $f_K = 0; $f_M = 0; $f_F = 0;
    $f_P = 0; $f_S = 0; $f_T = 0; $f_W = 0; $f_Y = 0; $f_V = 0;

    #output the sequence description
    my $desc = $FASTA_seq->display_id;
    #trim off possible trailing comma and whitespace
    $desc =~ s/,//g;
    $desc =~ s/\s*//g;

    print AACOUNTS $desc."> ";

    #get the sequence as an upper-case string
    my $sequence = uc $FASTA_seq->seq;

    #get the count of nucleotides
    $n_A = ($sequence =~ tr/A//); $n_R = ($sequence =~ tr/R//);
    $n_N = ($sequence =~ tr/N//); $n_D = ($sequence =~ tr/D//);
    $n_C = ($sequence =~ tr/C//); $n_E = ($sequence =~ tr/E//);
    $n_Q = ($sequence =~ tr/Q//); $n_G = ($sequence =~ tr/G//);
    $n_H = ($sequence =~ tr/H//); $n_I = ($sequence =~ tr/I//);
    $n_L = ($sequence =~ tr/L//); $n_K = ($sequence =~ tr/K//);
    $n_M = ($sequence =~ tr/M//); $n_F = ($sequence =~ tr/F//);
    $n_P = ($sequence =~ tr/P//); $n_S = ($sequence =~ tr/S//);
    $n_T = ($sequence =~ tr/T//); $n_W = ($sequence =~ tr/W//);
    $n_Y = ($sequence =~ tr/Y//); $n_V = ($sequence =~ tr/V//);

    #sum up for total
    $n_AA_Total = $n_A + $n_R + $n_N + $n_D + $n_C + $n_E + $n_Q + $n_G +
      $n_H + $n_I + $n_L + $n_K + $n_M + $n_F + $n_P + $n_S + $n_T +
```

```
$n_W + $n_Y + $n_V;

#calculate frequencies
$f_A = $n_A / $n_AA_Total;$f_R = $n_R / $n_AA_Total;
$f_N = $n_N / $n_AA_Total;$f_D = $n_D / $n_AA_Total;
$f_C = $n_C / $n_AA_Total; $f_E = $n_E / $n_AA_Total;
$f_Q = $n_Q / $n_AA_Total;$f_G = $n_G / $n_AA_Total;
$f_H = $n_H / $n_AA_Total;$f_I = $n_I / $n_AA_Total;
$f_L = $n_L / $n_AA_Total; $f_K = $n_K / $n_AA_Total;
$f_M = $n_M / $n_AA_Total;          $f_F = $n_F / $n_AA_Total;
$f_P = $n_P / $n_AA_Total; $f_S = $n_S / $n_AA_Total;
$f_T = $n_T / $n_AA_Total; $f_W = $n_W / $n_AA_Total;
$f_Y = $n_Y / $n_AA_Total;$f_V = $n_V / $n_AA_Total;

#round frequencies to six decimal places
$f_A = sprintf("%.3f", $f_A);$f_R = sprintf("%.3f", $f_R);
$f_N = sprintf("%.3f", $f_N);$f_D = sprintf("%.3f", $f_D);
$f_C = sprintf("%.3f", $f_C); $f_E = sprintf("%.3f", $f_E);
$f_Q = sprintf("%.3f", $f_Q);$f_G = sprintf("%.3f", $f_G);
$f_H = sprintf("%.3f", $f_H);$f_I = sprintf("%.3f", $f_I);
$f_L = sprintf("%.3f", $f_L); $f_K = sprintf("%.3f", $f_K);
$f_M = sprintf("%.3f", $f_M);          $f_F = sprintf("%.3f", $f_F);
$f_P = sprintf("%.3f", $f_P); $f_S = sprintf("%.3f", $f_S);
$f_T = sprintf("%.3f", $f_T); $f_W = sprintf("%.3f", $f_W);
$f_Y = sprintf("%.3f", $f_Y);$f_V = sprintf("%.3f", $f_V);

#write results to file, counts first then frequencies
print AACOUNTS "\nA(neutral): $n_A \t $f_A\n";
print AACOUNTS "R(BASIC): $n_R \t $f_R\n";
print AACOUNTS "N(neutral): $n_N \t $f_N\n";
print AACOUNTS "D(ACIDIC): $n_D \t $f_D\n";
print AACOUNTS "C(neutral): $n_C \t $f_C\n";
print AACOUNTS "E(ACIDIC): $n_E \t $f_E\n";
print AACOUNTS "Q(neutral): $n_Q \t $f_Q\n";
print AACOUNTS "G(neutral): $n_G \t $f_G\n";
print AACOUNTS "H(BASIC): $n_H \t $f_H\n";
print AACOUNTS "I(neutral): $n_I \t $f_I\n";
print AACOUNTS "L(neutral): $n_L \t $f_L\n";
print AACOUNTS "K(BASIC): $n_K \t $f_K\n";
print AACOUNTS "M(neutral): $n_M \t $f_M\n";
print AACOUNTS "F(neutral): $n_F \t $f_F\n";
print AACOUNTS "P(neutral): $n_P \t $f_P\n";
print AACOUNTS "S(neutral): $n_S \t $f_S\n";
print AACOUNTS "T(neutral): $n_T \t $f_T\n";
print AACOUNTS "W(neutral): $n_W \t $f_W\n";
print AACOUNTS "Y(neutral): $n_Y \t $f_Y\n";
```

```
        print AACOUNTS "V(neutral): $n_V \t $f_V\n";
        print AACOUNTS "Total: $n_AA_Total \n";
    }

    close AACOUNTS;
}
```

**changeCode.pl:**

```
#!/bin/perl -w
use strict;
use Bio::Seq;
use Bio::SeqIO;
use Bio::Tools::OddCodes;

# Matthew Conte
#
# This script converts the amino acids from the sequences in a FASTA file
# into the alphabet of the user's choice with the methods provided in
# in Bio::Tools::OddCodes.
#
#
# Output is to FASTAfilename.FASTA
#
# Usage: perl changeCode.pl file.FASTA file2.FASTA ...

foreach my $file(@ARGV) {
    print STDERR "Reading input file $file... \n";

    #open the FASTA file
    my $FASTA_in = Bio::SeqIO->new(-file => $file,
                                   -format => 'FASTA');

    #open the file's basename
    $file =~ s/\.seq$//g;

    #open the .charge file for writing
    #NOTE - change .charge to .oddcode for whichever oddcode you decide to
    #use.  Options are: charge, chemical, functional, hydrophobic.
    open CHARGE_COUNTS, ">$file.charge";

    #print a line at the top so that dipeps.pl understands that it is a long
    #FASTA sequence.
```

```
    print CHARGE_COUNTS ">gi|our long sequence\n";

    #for each sequence in the FASTA file..
    while(my $FASTA_seq = $FASTA_in->next_seq()){

        #change the sequence in the file to the alphabet of your choosing
        #in this case chemical is chosen
        #Options are: charge, chemical, functional, hydrophobic.

        my $oddcode_obj = Bio::Tools::OddCodes->new(-seq => $FASTA_seq);
        my $sequence = $oddcode_obj->charge();

        print CHARGE_COUNTS $$sequence;

    }
    close CHARGE_COUNTS;
}
```

**charge.pl:**

```
#!/bin/perl -w
use strict;
use Bio::Seq;
use Bio::SeqIO;
use Bio::Tools::OddCodes;

# Matthew Conte
#
# This script converts the amino acids from the sequences in a FASTA file into
# a 3-letter alphabet using the charge() method in Bio::Tools::OddCodes.
# It then counts the number of each code for each sequence as well as each
# frequency.
#
# Alphabet: A (negatively), C (positively), N (no charge).
#
# Output is to FASTAfilename.charge_counts
# Output is TAB-DELIMITED for import in to Microsoft Excel
#
# Usage: perl charge.pl file.FASTA file2.FASTA ...

#initialize count variables
my $n_A = 0;
my $n_C = 0;
```

```
my $n_N = 0;
my $n_AA_Total = 0;

#initialize frequency variables
my $f_A = 0;
my $f_C = 0;
my $f_N = 0;

foreach my $file(@ARGV) {
   print STDERR "Reading input file $file... \n";

   #open the FASTA file
   my $FASTA_in = Bio::SeqIO->new(-file => $file,
                                  -format => 'FASTA');

   #open the file's basename
   $file =~ s/\.seq$//g;

   #open the .chem_counts file for writing
   open CHARGE_COUNTS, ">$file.charge_counts";

   #top line in the file to see where everything goes
   print CHARGE_COUNTS
"Sequence\tA(positive)\t%A(positive)\tC(negative)\t%C(negative)\tN(no charge)\t%N(no
charge)\tTotal\n";

   #for each sequence in the FASTA file..
   while(my $FASTA_seq = $FASTA_in->next_seq()){

        #reset variables
        $n_A = 0; $n_C = 0; $n_N = 0;
        $n_AA_Total = 0;
        $f_A = 0; $f_C = 0; $f_N = 0;

        #output the sequence description
        my $desc = $FASTA_seq->display_id;
        #trim off possible trailing comma and whitespace
        $desc =~ s/,//g;
        $desc =~ s/\s*//g;

        print CHARGE_COUNTS $desc;

        my $oddcode_obj = Bio::Tools::OddCodes->new(-seq => $FASTA_seq);
        my $sequence = $oddcode_obj->charge();

        ##get the count of amino acids
```

```
        $n_A = ($$sequence =~ tr/A//);
        $n_C = ($$sequence =~ tr/C//);
        $n_N = ($$sequence =~ tr/N//);

        #sum up for total
        $n_AA_Total = $n_A + $n_C + $n_N;

        #calculate frequencies
        $f_A = $n_A / $n_AA_Total;
        $f_C = $n_C / $n_AA_Total;
        $f_N = $n_N / $n_AA_Total;

        #round frequencies to 3 decimal places
        $f_A = sprintf("%.3f", $f_A);
        $f_C = sprintf("%.3f", $f_C);
        $f_N = sprintf("%.3f", $f_N);

        #write results to file
        print  CHARGE_COUNTS
"\t$n_A\t$f_A\t$n_C\t$f_C\t$n_N\t$f_N\t$n_AA_Total\n";

    }

    close CHARGE_COUNTS;
}
```

**chemical.pl:**

```
#!/bin/perl -w
use strict;
use Bio::Seq;
use Bio::SeqIO;
use Bio::Tools::OddCodes;

# Matthew Conte
#
# This script converts the amino acids from the sequences in a FASTA file into
# a 8-letter alphabet using the chemical() method in Bio::Tools::OddCodes.
# It then counts the number of each code for each sequence as well as each
# frequency.
#
# Alphabet: A (acidic), L (aliphatic), M (amide), R (aromatic), C (basic),
#         H (hydroxyl), I (imino), S (sulphur).
#
```

```
# Output is to FASTAfilename.chem_counts
# Output is TAB-DELIMITED for import in to Microsoft Excel
#
# Usage: perl chemical_AA_counts.pl file.FASTA file2.FASTA ...

#initialize count variables
my $n_A = 0;
my $n_L = 0;
my $n_M = 0;
my $n_R = 0;
my $n_C = 0;
my $n_H = 0;
my $n_I = 0;
my $n_S = 0;
my $n_AA_Total = 0;

#initialize frequency variables
my $f_A = 0;
my $f_L = 0;
my $f_M = 0;
my $f_R = 0;
my $f_C = 0;
my $f_H = 0;
my $f_I = 0;
my $f_S = 0;

foreach my $file(@ARGV) {
   print STDERR "Reading input file $file... \n";

   #open the FASTA file
   my $FASTA_in = Bio::SeqIO->new(-file => $file,
                                  -format => 'FASTA');

   #open the file's basename
   $file =~ s/\.seq$//g;

   #open the .chem_counts file for writing
   open CHEM_COUNTS, ">$file.chem_counts";

   #top line in the file to see where everything goes
   print CHEM_COUNTS
"Sequence\tA(acidic)\t%A(acidic)\tL(aliphatic)\t%L(aliphatic)\tM(amide)\t%M(amide)\t
R(aromatic)\t%R(aromatic)\tC(basic)\t%C(basic)\tH(hydroxyl)\t%H(hydroxyl)\tI(imino)
t%I(imino)\tS(sulphur)\t%S(sulphur)\tTotal\n";

   #for each sequence in the FASTA file..
```

```
while(my $FASTA_seq = $FASTA_in->next_seq()){

    #reset variables
    $n_A = 0; $n_L = 0; $n_M = 0; $n_R = 0; $n_C = 0; $n_H = 0; $n_I = 0;
    $n_S = 0;
    $n_AA_Total = 0;
    $f_A = 0; $f_L = 0; $f_M = 0; $f_R = 0; $f_C = 0; $f_H = 0; $f_I = 0;
    $f_S = 0;

    #output the sequence description
    my $desc = $FASTA_seq->display_id;
    #trim off possible trailing comma and whitespace
    $desc =~ s/,//g;
    $desc =~ s/\s*//g;

    print CHEM_COUNTS $desc;

    my $oddcode_obj = Bio::Tools::OddCodes->new(-seq => $FASTA_seq);
    my $sequence = $oddcode_obj->chemical();

    ##get the count of amino acids
    $n_A = ($$sequence =~ tr/A//);
    $n_L = ($$sequence =~ tr/L//);
    $n_M = ($$sequence =~ tr/M//);
    $n_R = ($$sequence =~ tr/R//);
    $n_C = ($$sequence =~ tr/C//);
    $n_H = ($$sequence =~ tr/H//);
    $n_I = ($$sequence =~ tr/I//);
    $n_S = ($$sequence =~ tr/S//);

    #sum up for total
    $n_AA_Total = $n_A + $n_L + $n_M + $n_R + $n_C + $n_H + $n_I +$n_S;

    #calculate frequencies
    $f_A = $n_A / $n_AA_Total;
    $f_L = $n_L / $n_AA_Total;
    $f_M = $n_M / $n_AA_Total;
    $f_R = $n_R / $n_AA_Total;
    $f_C = $n_C / $n_AA_Total;
    $f_H = $n_H / $n_AA_Total;
    $f_I = $n_I / $n_AA_Total;
    $f_S = $n_S / $n_AA_Total;

    #round frequencies to 3 decimal places
    $f_A = sprintf("%.3f", $f_A);
    $f_L = sprintf("%.3f", $f_L);
```

```
        $f_M = sprintf("%.3f", $f_M);
        $f_R = sprintf("%.3f", $f_R);
        $f_C = sprintf("%.3f", $f_C);
        $f_H = sprintf("%.3f", $f_H);
        $f_I = sprintf("%.3f", $f_I);
        $f_S = sprintf("%.3f", $f_S);

        #write results to file
        print  CHEM_COUNTS
"\t$n_A\t$f_A\t$n_L\t$f_L\t$n_M\t$f_M\t$n_R\t$f_R\t$n_C\t$f_C\t$n_H\t$f_H\t$n_I\t
$f_I\t$n_S\t$f_S\t$n_AA_Total\n";

   }

   close CHEM_COUNTS;
}
```

**dipeps.pl:**

```
#!/bin/perl -w
use strict;
use Bio::Seq;
use Bio::SeqIO;
use Bio::Tools::OddCodes;
use Bio::Tools::SeqWords;

# Matthew Conte
#
# This script counts the number of each different amino acid pair for each
# sequence in the given FASTA files
#
#
# Output is to FASTAfilename.dipep_counts
# Output is TAB-DELIMITED for import in to Microsoft Excel
#
# Usage: perl dipep.pl file.FASTA file2.FASTA ...

# variable to count the total number of amino acids in each sequence.
my $total = 0;

foreach my $file(@ARGV) {
   print STDERR "Reading input file $file... \n";

   #open the FASTA file
```

```
my $FASTA_in = Bio::SeqIO->new(-file => $file,
                               -format => 'FASTA');

#open the file's basename
$file =~ s/\.seq$//g;

#open the .func_counts file for writing
open DIPEP_COUNTS, ">$file.dipep_counts";


#for each sequence in the FASTA file..
while(my $FASTA_seq = $FASTA_in->next_seq()){

    #reset total
    $total = 0;

    #output the sequence description
    my $desc = $FASTA_seq->display_id;
    #trim off possible trailing comma and whitespace
    $desc =~ s/,//g;
    $desc =~ s/\s*//g;

    print DIPEP_COUNTS "\n$desc";

    my $seq_word = Bio::Tools::SeqWords->new(-seq => $FASTA_seq);
    my $sequence = $seq_word->count_overlap_words(2);

    # display the hashtable
    my %hash = %$sequence;

    #this code will sort the dipeptides alphabetically

    #foreach my $key(sort keys %hash){
      #$total = $total + $hash{$key};
      #print DIPEP_COUNTS "\n$key\t$hash{$key}";
    #}

    # sort the hash by value in descending order (highest to lowest)
    foreach my $key (sort {$hash{$b} cmp $hash{$a} } keys %hash){
      $total = $total + $hash{$key};
      print DIPEP_COUNTS "\n$key\t$hash{$key}";
    }
    print DIPEP_COUNTS "\nTotal: $total";
}

close DIPEP_COUNTS;
```

}

**dipepsA.pl:**

```
#!/bin/perl -w
use strict;
use Bio::Seq;
use Bio::SeqIO;
use Bio::Tools::OddCodes;
use Bio::Tools::SeqWords;

# Matthew Conte
#
# This script counts the number of each different amino acid pair for each
# sequence in the given FASTA files
#
# Output is sorted alphabetically by amino acid pair
# Output is to FASTAfilename.dipep_counts
# Output is TAB-DELIMITED for import in to Microsoft Excel
#
# Usage: perl dipep.pl file.FASTA file2.FASTA ...


# variable to count the total number of amino acids in each sequence.
my $total = 0;

foreach my $file(@ARGV) {
   print STDERR "Reading input file $file... \n";

   #open the FASTA file
   my $FASTA_in = Bio::SeqIO->new(-file => $file,
                                  -format => 'FASTA');

   #open the file's basename
   $file =~ s/\.seq$//g;

   #open the .func_counts file for writing
   open DIPEP_COUNTS, ">$file.dipepA_counts";


   #for each sequence in the FASTA file..
   while(my $FASTA_seq = $FASTA_in->next_seq()){

        #reset total
```

```
        $total = 0;

        #output the sequence description
        my $desc = $FASTA_seq->display_id;
        #trim off possible trailing comma and whitespace
        $desc =~ s/,//g;
        $desc =~ s/\s*//g;

        print DIPEP_COUNTS "\n$desc";

        my $seq_word = Bio::Tools::SeqWords->new(-seq => $FASTA_seq);
        my $sequence = $seq_word->count_overlap_words(2);

        # display the hashtable
        my %hash = %$sequence;

        #this code will sort the dipeptides alphabetically

        foreach my $key(sort keys %hash){
           $total = $total + $hash{$key};
           print DIPEP_COUNTS "\n$key\t$hash{$key}";
        }

        # sort the hash by value in descending order (highest to lowest)
        #foreach my $key (sort {$hash{$b} cmp $hash{$a} } keys %hash){
           #$total = $total + $hash{$key};
           #print DIPEP_COUNTS "\n$key\t$hash{$key}";
        #}
        print DIPEP_COUNTS "\nTotal: $total";
   }

   close DIPEP_COUNTS;
}
```

**functional.pl:**

```
#!/bin/perl -w
use strict;
use Bio::Seq;
use Bio::SeqIO;
use Bio::Tools::OddCodes;

# Matthew Conte
#
```

```
# This script converts the amino acids from the sequences in a FASTA file into
# a 4-letter alphabet using the functional() method in Bio::Tools::OddCodes.
# It then counts the number of each code for each sequence as well as each
# frequency.
#
# Alphabet: A (acidic), C (basic), H (hydrophobic), P (polar).
#
# Output is to FASTAfilename.funct_counts
# Output is TAB-DELIMITED for import in to Microsoft Excel
#
# Usage: perl functionalcounts.pl file.FASTA file2.FASTA ...

#initialize count variables
my $n_A = 0;
my $n_C = 0;
my $n_H = 0;
my $n_P = 0;
my $n_AA_Total = 0;

#initialize frequency variables
my $f_A = 0;
my $f_C = 0;
my $f_H = 0;
my $f_P = 0;

foreach my $file(@ARGV) {
   print STDERR "Reading input file $file...\n";

   #open the FASTA file
   my $FASTA_in = Bio::SeqIO->new(-file => $file,
                                  -format => 'FASTA');

   #open the file's basename
   $file =~ s/\.seq$//g;

   #open the .func_counts file for writing
   open FUNC_COUNTS, ">$file.func_counts";

   #top line in the file to see where everything goes
   print FUNC_COUNTS
"Sequence\tA(Acidic)\t%A(Acidic)\tC(Basic)\t%C(Basic)\tH(Hydrophobic)\t%H(Hydro
phobic)\tP(Polar)\t%P(Polar)\tTotal\n";

   #for each sequence in the FASTA file..
   while(my $FASTA_seq = $FASTA_in->next_seq()) {
```

```
        #reset variables
        $n_A = 0; $n_C = 0; $n_H = 0; $n_P = 0;
        $n_AA_Total = 0;
        $f_A = 0; $f_C = 0; $f_H = 0; $f_P = 0;

        #output the sequence description
        my $desc = $FASTA_seq->display_id;
        #trim off possible trailing comma and whitespace
        $desc =~ s/,//g;
        $desc =~ s/\s*//g;

        print FUNC_COUNTS $desc;

        my $oddcode_obj = Bio::Tools::OddCodes->new(-seq => $FASTA_seq);
        my $sequence = $oddcode_obj->functional();

        ##get the count of amino acids
        $n_A = ($$sequence =~ tr/A//);
        $n_C = ($$sequence =~ tr/C//);
        $n_H = ($$sequence =~ tr/H//);
        $n_P = ($$sequence =~ tr/P//);

        #sum up for total
        $n_AA_Total = $n_A + $n_C + $n_H + $n_P;

        #calculate frequencies
        $f_A = $n_A / $n_AA_Total;
        $f_C = $n_C / $n_AA_Total;
        $f_H = $n_H / $n_AA_Total;
        $f_P = $n_P / $n_AA_Total;

        #round frequencies to 3 decimal places
        $f_A = sprintf("%.3f", $f_A);
        $f_C = sprintf("%.3f", $f_C);
        $f_H = sprintf("%.3f", $f_H);
        $f_P = sprintf("%.3f", $f_P);

        #write results to file
        print FUNC_COUNTS
"\t$n_A\t$f_A\t$n_C\t$f_C\t$n_H\t$f_H\t$n_P\t$f_P\t$n_AA_Total\n";

    }

    close FUNC_COUNTS;
}
```

**hydro.pl:**

```
#!/bin/perl -w
use strict;
use Bio::Seq;
use Bio::SeqIO;
use Bio::Tools::OddCodes;

# Matthew Conte
#
# This script converts the amino acids from the sequences in a FASTA file into
# a 2-letter alphabet using the hydrophobic() method in Bio::Tools::OddCodes.
# It then counts the number of each code for each sequence as well as each
# frequency.
#
# Alphabet: I (hydrophilic), O (hydrophobic).
#
# Output is to FASTAfilename.hydro_counts
# Output is TAB-DELIMITED for import in to Microsoft Excel
#
# Usage: perl hydro.pl file.FASTA file2.FASTA ...

#initialize count variables
my $n_I = 0;
my $n_O = 0;
my $n_AA_Total = 0;

#initialize frequency variables
my $f_I = 0;
my $f_O = 0;

foreach my $file(@ARGV) {
   print STDERR "Reading input file $file... \n";

   #open the FASTA file
   my $FASTA_in = Bio::SeqIO->new(-file => $file,
                                  -format => 'FASTA');

   #open the file's basename
   $file =~ s/\.seq$//g;

   #open the .chem_counts file for writing
   open HYDRO_COUNTS, ">$file.hydro_counts";
```

```
    #top line in the file to see where everything goes
    print HYDRO_COUNTS
"Sequence\tI(hydrophilic)\t%I(hydrophilic)\tO(hydrophobic)\t%O(hydrophobic)\tTotal\n
";

    #for each sequence in the FASTA file..
    while(my $FASTA_seq = $FASTA_in->next_seq()){

        #reset variables
        $n_I = 0; $n_O = 0;
        $n_AA_Total = 0;
        $f_I = 0; $f_O = 0;

        #output the sequence description
        my $desc = $FASTA_seq->display_id;
        #trim off possible trailing comma and whitespace
        $desc =~ s/,//g;
        $desc =~ s/\s*//g;

        print HYDRO_COUNTS $desc;

        my $oddcode_obj = Bio::Tools::OddCodes->new(-seq => $FASTA_seq);
        my $sequence = $oddcode_obj->hydrophobic();

        ##get the count of amino acids
        $n_I = ($$sequence =~ tr/I//);
        $n_O = ($$sequence =~ tr/O//);

        #sum up for total
        $n_AA_Total = $n_I + $n_O;

        #calculate frequencies
        $f_I = $n_I / $n_AA_Total;
        $f_O = $n_O / $n_AA_Total;

        #round frequencies to 3 decimal places
        $f_I = sprintf("%.3f", $f_I);
        $f_O = sprintf("%.3f", $f_O);

        #write results to file
        print  HYDRO_COUNTS "\t$n_I\t$f_I\t$n_O\t$f_O\t$n_AA_Total\n";

    }

    close HYDRO_COUNTS;
}
```

**makeComposite.pl:**

```
#!/bin/perl
use strict;

# Matthew Conte
#
# This script converts FASTA files of multiple sequences into a single (composite)
# sequence.  This composite sequence is then able to be used with other programs
# such as charge.pl, chemical.pl, dipeps.pl, functional.pl, and hydro.pl.
#
# Usage: perl makeComposite.pl file.FASTA > outfile

my $count = 0;

while(<>){
   if ($count < 1) {
         $_;
   } else {

         if (/^>gi*/) {
            #do nothing
         } else {
            print $_;
         }
   }
   $count++;
}
print "$count\n";
```